MASTER IN COMPUTER SCIENCE

Ontological evaluation of BMM and i* with the UEML approach

Tu, Christophe

Award date:
2007

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique

Ontological evaluation of BMM and i* with the UEML approach

Christophe TU

Mémoire présenté en vue de l'obtention du grade de Maître en Informatique
Année academique 2006-2007

Abstract

Confronted with a changing economical environment, enterprises are compelled to be more flexible and pro-active in order to be able to anticipate and to adapt to frequent changes they have to face. Mastering changes require enterprises to determine, as clearly as possible, their operation mode and their surrounding environment. Entreprise Modelling (EM) is an essential tool to carry out this task. The broad scope of EM led to many different Enterprise Modelling Languages (EMLs). As a consequence, a genuine need for EMLs interoperability arose as more and more cooperating enterprises have to deal with many different incompatible but interrelated models.
Unified Enterprise Modelling language (UEML) aims to establish itself as a solution to this need by creating a federator language. Within the framework of this project, a method has been developed in order to analyze EMLs and to integrate the ensuing knowledge into an ontology. This work aims at enhancing the currrent UEML version by analysing and intergrating two EMLs (BMM and i^*) using the UEML 2.0 approach.

Résumé

Confrontées à un environnement économique changeant, les entreprises sont contraintes à être plus flexibles et plus proactives afin de pouvoir anticiper et s'adapter aux changements fréquents auxquels elles doivent faire face. La maîtrise des changements requiert des entreprises qu'elles déterminent, de manière la plus précise possible, leur mode de fonctionnement et l'environnement qui les entoure. Entreprise Modelling (EM) est un outil essentiel pour réaliser cette tâche. Le large champ d'application de EM a conduit à différents Enterprise Modelling Languages (EMLs). Un réel besoin d'interopérabilité des EMLs a vu le jour étant donné que les entreprises, coopérant de plus en plus entre elles, doivent faire face à de nombreux modèles incompatibles mais étroitement liés.
Unified Enterprise Modelling Language (UEML) vise à s'imposer comme la solution à ce besoin en créant un langage fédérateur. Dans le cadre de ce projet, une méthode a été développée pour analyser les EMLs et pour intégrer les connaissances qui en découlent dans une ontologie. Ce travail a pour objectif de contribuer à l'enrichissement de la version actuelle de l'UEML en analysant et intégrant deux langages de modélidation d'entreprise (BMM and i^*) en utilisant l'approche UEML 2.0.

Preface

Writing this thesis gave me an excellent opportunity to deepen an interesting research topic and also to experience the Italian culture.
This work has been mainly carried out during the first semester of the academic year 2006/2007 at University of Torino under the supervision of Pr. Giuseppe BERIO and finalized at University of Namur.
Pr. BERIO is one of the main actors of the UML 2.0 project, a Network of Excellence supported by the European Commission.
I would like to express my gratitude to Pr. BERIO for his warm welcome, his guidance, availability and support during my internship in Torino. It has been an awesome experience.
I am deeply indebted to my promotor, Pr. Michaël PETIT for introducing me to the topic, for assisting me during the progress of my work and for giving me the opportunity to live a thrilling experience abroad. Your stimulating suggestions and encouragement helped me in all the time of research and writing of this thesis.
I would also like to thank Doctor Raimundas MATULEVICIUS who monitored my work and took effort in reading and providing me with valuable comments on earlier versions of this thesis. He stood by me adding valuable advice and comments throughout my work, which enhanced my thesis as well as my research learning process.
Lastly, I wish to thank my parents and Daphné for their loving support, their encouragement and understanding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

The current economical environment is subject to frequent changes. In order to anticipate and to adapt to this changing environment, entreprises are compelled to be more flexible and pro-active. The entreprise structure can be deeply impacted by these changes of technological, sociological or economical nature. For enterprises working in a competitive area, mastering these changes can be a crucial success factor.
For that purpose, entreprises have to assess as precisely as possible their operation mode and the surrounding environment. Entreprise Modelling (EM) is one of the tools enables them to achieve this task. EM makes possible externalization of information regarding many facets of the entreprise, for instance description of its organization or its operational processes. The scope is very broad; it can be used to analyze and to restructure enterprises in order to improve results, to make a comparison between different possible scenarios leading to the best solution or to train new employees cooperating with the company.

Every sector, ranging from goods producers to services providers, can use EM. The use of EM is also possible to model different aspects of entreprises such as organization, resources, process, information, requirements, goals or strategy. This extremely large scope gave birth to many different Entreprise Modelling Languages (EML). Inside one entreprise, the different enterpise models are interconnected interrelated. However, most EMLs are implemented in tools with proprietary terminology and modelling constructs. Therefore, the various enterprise models are incompatible. Consequently, a genuine need for EMLs interoperability arose.

Unified Enterprise Modelling Language (UEML) aims to establish itself as a solution to this need. Two projects, UEML 1.0 and UEML 2.0, were

set up in order to create a federator language that enables to integrate existing modelling languages and to support their comparison, consistency checking, update reflection, view synchronization and, eventually, model-to-model translation across modelling language boundaries. The two projects are akin but differ with regard to their goals and approaches. The first one is made up of three different languages while the second one aims to integrate new languages. The second project analyzes the different EMLs by etablishing an ontology about what they represent in the real world.

The main goal of our work is to contribute to the enhancement of the UEML 2.0 by analysing two EMLs: the Business Motivation Model (BMM) and the i^* framework. The analysis will be performed using the UEML 2.0 approach.

1.2 About the document

The document is divided into three parts. The first part explains the background of the analysis. Chapter 2 gives us an insight into the EMLs and emphasizes the need for interoperability. Chapter 3 defines UEML, briefly explains the UEML 1.0 project and details UEML 2.0 project with a particular emphasis on the UEML approach. Chapter 4 describes various UEML tools while chapter 5 focuses on the 3 EMLs that will be used in our analysis (BMM and i^*) and in the evaluation (GRL).

The contribution is the second part of the document and starts in chapter 6 with a description of two methodologies we will be using for our analysis: chapter 7 for the ontological analysis of BMM and chapter 8 for the ontological analysis of i^*.

The third part will be an evaluation and will present a constrained valildation of BMM and i^* carried out by the use of the UEML Validator tool (chapter 9). In chapter 10, we will evaluate an UEML tool called Similarity Plug-In.

Finally, we will conclude our work in chapter 11 by giving its future developments.

Part I

Background

Chapter 2

Enterprise Modelling Languages

This chapter provides a general overview of the context in which Enterprise Modelling appeared and demonstrates the necessity for today's enterprises to use Enterprise Modelling in order to stay competitive in a global economy context. The chapter also defines the drawbacks for companies to confront with several Enterprise Modelling Languages and underlines the needs for interoperability and sharable enterprise knowledge.

2.1 Enterprise Modelling

2.1.1 The context

Nowadays, companies are evolving in a fast changing world. The evolution of technology has completely modified the environment in which enterprises evolve. Indeed, by influencing economic, politic and social aspects of the environment, the spread of technology has been one of the major factor which has permit to the economy to become global. It also gave birth to a brand new concept: the global economy.

The global economy's main characteristic is called the globalization. This concept can be defined as follows [SI03]:

"Globalization is the worldwide process which makes prices, products, wages, rates of interest and profits become almost the same everywhere. Globalization uses three forces for development: the role of human migration, international trade, and rapid movements of capital and integration of financial markets."

The global economy give business the ability to market products and

services all over the globe. It also allows them to develop partnerships and alliances throughout the world, which has become essential for success in today's business. This increase in globalization has created many new opportunities, such as niche markets, and requires everyone to keep up with globalization in order to stay competitive.

As [Shane] points out:

"Technology and trade separate the economy into two camps: those with the skills to participate in the global economy and those who lack them."

2.1.2 Enterprise Modelling in a global economy

In order to play a significant role or at least to survive in this global economy, many business companies need to have a clear vision of their own structure. They also need to understand, control and decide the way they operate in order to be able to face efficiently changes imposed by their environment and to fulfil their commitments in terms of quality, cost, delay and reactivity. Thus, an efficient design, analysis and optimization of enterprise operations require notations, formalisms, methods and tools to describe the various facets of a business organization. The aspects of this modelling process are called Enterprise Modelling (EM).

In a more formal sense, EM can be characterized by the following definition [Petit]:

"EM is the set of activities or process used to develop the various parts of an enterprise model to address some desired modelling finality. It can also be defined as the art of "externalising" enterprise knowledge, i.e. representing the enterprise in terms of its organisation and operations (e.g. processes, behaviour, acivities, information, object and material flows, resources and organisation units, system infrastructure and architectures)."

The main goal of EM is therefore to support enterprise analysis. Enterprise analysis are principally directed towards enterprise operations, capitalizations of acquired knowledge, enterprise design, descriptions of various aspects of enterprises, enterprise behaviour simulations, support for descisions of enterprise operations or control, coordination and monitoring of some parts of the enterprise.

Thus we can sum up the major uses of EM in the four following points. EM can be used as:

- a decision support for evaluating operational alternatives;
- a communication tool that enables the mutual understanding of issues between stakeholders of the enterprise, both internal and external ones;
- a model driven operation control and monitoring, for efficient business process execution;
- a training of new personnel, where enterprise models serve as demonstration of the real business process for new employees.

2.2 Entreprise Modelling Languages

2.2.1 Definition

Enterprise Modelling Languages (EMLs) are the key elements to develop EM. According to GERAM (Generalised Enterprise Reference Architecture and Methodology) [Petit], EMLs can be defined as follows:

"EMLs define the generic modelling constructs for enterprise modelling adapted to the needs of people creating and using enterprise models. In particular enterprise modelling languages will provide construct to describe and model human roles, operational processes and their functional contents as well as the supporting information, office and production technologies."

2.2.2 Numerous EMLs

During the 80s, Europe started to launch some Enterprise Modelling projects which led to the birth of several EMLs, including among others GRAI and CIMOSA.
This situation became worse during the 90s. Indeed, more and more EMLs (such as IDEF3, IEM or DEM) and EM tools (such as ARIS Toolset, FirstSTEP, NCR Metis, Bonapart, Enterprise Modeller, PROPLAN, PrimeObject, MOOGO, CimTool or IMAGIM) appeared. The marketplace became flooded with all those different EM techniques and tools.

The huge number of existing EML's created a Tower of Babel situation for business users interested in using EM. According to [Ver02], this Tower of Babel situation can be summarized by the following facts:

- there were too many EM languages to learn and to understand, as well as too many EM tools with completely different interfaces,
- there was instability of vocabulary and of modelling paradigms (within two methods, the same concept may have different names and happens

to be modelled differently, while the same term may refer to different things),

- there were many incompatible EM tools on the marketplace, which are not able to inter-operate and which can hardly exchange models,
- there were no, or poor, formal foundations for EM.

2.2.3 Needs for interoperability

Many EM technologies available on the market offered efficient but different functionalities and semantics. They were rarely able to interoperate and to communicate. The resulting enterprise models were often incompatible and they prohibited the interoperability of working solutions. Modelling tools talked about the same things but did not talk to one another. A common standard core language and some common extension mechanisms were missing, making the sharing of models across tools impossible. This was a serious drawback for awareness, acceptance and wide use of the EM technology in industry. At that time, business enterprises were hesitant to trust and to invest in EM technologies because they were not sure to capitalize on the modelling efforts they could have consented. This situation really hindered entreprise integration, interoperability, and sharable enterprise knowledge. However, EM techniques and associated visual languages had become indispensable to support new approaches to business transformation and improvement, in developing smart organisations and networked organisations. The industrial needs for a common standard core language became urgent. Those urgent needs for interoperability were fulfilled by the UEML project which is given over to researches to develop of a standardized Unified Enterprise Modelling Language.

2.3 Summary

In this chapter, we explored the context in which today's enterprises evolve. We pointed out how EM can help enteprises to face with global economy challenges. We emphazied the major probem of using EM for those enterprises: The lack of interoperability between EMLs which was created by the large number of EMLs available on the market place. In the next chapter we will introduce and explain how UEML provides a solution to interoperability.

Chapter 3

UEML

The previous chapter explains how companies reach a better internal integration by using different EMLs and tools. The integration efficiency depends on the extent of cooperation of the modelling languages. Unified Enterprise Modelling Language (UEML) is a solution to integrate them and requires the use of the least necessary number of translator. The current chapter gives an explanation on UEML, its goals and method. Two different versions of UEML (1.0 and 2.0) are available. On the whole, both have a common goal which is to support enterprise model exchange (integration, translation and transformation) and global consistency between evolving enterprise models. However, each has also different specific goals and consequently, adopt different approaches.

3.1 Introduction to UEML

3.1.1 UEML context

The idea of a Unified Enterprise Modelling Language (UEML) emerged in order to bring a solution to the "numerous EMLs" problem explained in chapter 2. The topic arose for the first time in 1997 during the International Conference of Enterprise Integration and Modelling Techniques conference (ICEIMT), with UEML aiming of being an intermediate language which would be able to incorporate and support integrated use of a wide variety of existing modelling languages. Since then, UEML has become a concrete achievement; at first with UEML 1.0 developed from 2002 to 2003 within the context of the UEML project and then with UEML 2.0 developed from 2004 until today within the context of the INTEROP project. This evolution of UEML will be explained further later in the chapter. But first, we give a proper definition of UEML.

3.1.2 UEML definition

UEML stands for Unified Enterprise Modelling Language. As underlined by [Ber03], UEML can be defined by interpreting each acronym in capital letter:

Unified means:

"unified and shared linguistic context for..."

Enterprise Modelling means:

"supporting all the needed tasks for representing and utilizing enterprise knowledge through a..."

Language means:

"with well-defined syntax and, possibly, semantics."

According to [Petit], the long term objective of UEML is the definition of a Unified Enterprise Modelling Language, which would serve as an interlingua between EM tools. This language will:

- Provide the business community with a common visual template based language to be used on top of most commercial enterprise modelling and workflow software tools;
- Provide standardized mechanisms for sharing knowledge models and exchanging enterprise models among projects, overcoming tool dependencies;
- Support the implementation of open and evolutionary enterprise model repositories to leverage enterprise knowledge engineering services and capabilities.

Figure 3.1 shows the principle of the UEML interoperability model: The tool 1 can work on customer model B via the gateway of a common UEML API and tool specific UEML API interpreters supported by the common repository.

3.1.3 Integration problems

Given the numerous EMLs available, we need to define some commonalities between constructs belonging to each language in order to obtain a commun representation of knowledge. Thus UEML comprise constructs common to the most representative existing EMLs. In the integration process, we need to take into account and to tackle the following problems [BAO04]:

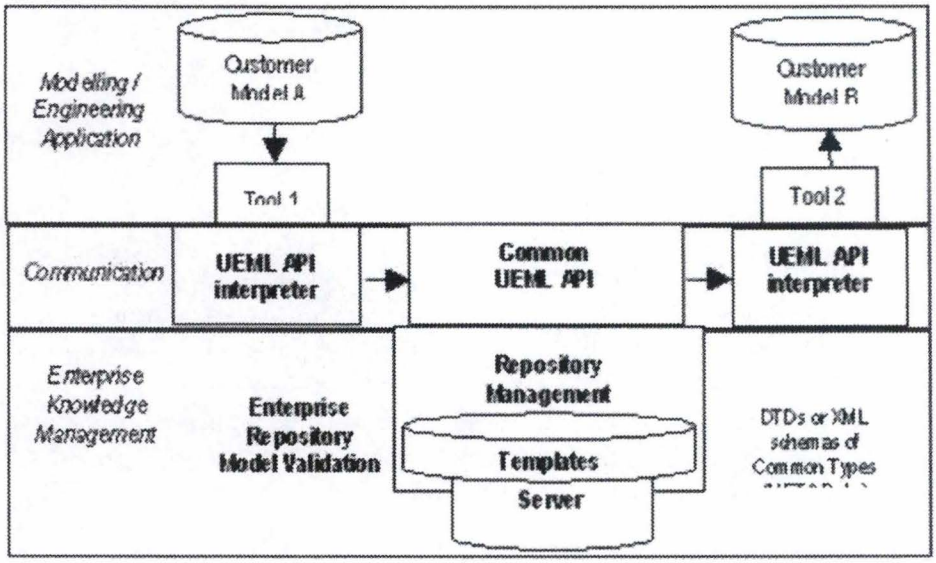

Figure 3.1: The UEML interoperability model (from [Dou02]).

- differences in the (abstract) syntax: even in a unique business domain, EMLs differ syntactically;
- coverage and expressivity of the Enterprise Modelling Languages: Enterprise Modelling Languages have different focuses and purposes;
- differences in semantics of similar constructs: sometimes enterprises use the same constructs but they associate distinct meanings.

UEML handles these integration problems within the language dimension, in the following ways:

Differences in the (abstract) syntax: UEML is founded in the agreement concerning the structure (i.e. the Constructs) of the knowledge (i.e. the Concepts underlying the Constructs) being exchanged or translated. This (abstract) syntax is expressed through a class model and specifically (just for convenience) a UML class model.

Coverage and expressivity of the Enterprise Modelling Languages: the decision to be taken is related to what kind, and to which extent knowledge should be shared by using UEML; in this case, the following equation has been defined:

$$UEML = CommonConcepts + (someof)NonCommonConcepts$$

This equation well represents the UEML as a federator containing what it is shared among various languages.

Differences in semantics of "similar" constructs: it is the most difficult type of problem and it is related to the definition of common concepts. A part of the complexity is due to the fact that the underlying concepts of most EMLs constructs are provided by text expressed in natural language which is ambiguous. Using these texts for finding semantic relationships between constructs belonging to distinct languages is thus very difficult and quite subjective. However, database integration suggests to use examples to cope with that kind of problem. The purpose is to identify and verify clearly the "intentional relationships between concepts" by using the evidence of available examples of such concepts. In practice, if two tables of different schemata allow to represent the same instances, we can suppose that those tables are equivalent. An analogy can be done between languages and databases. Modelling languages artifacts play the role of database's instances, ML's model plays the role of the database and, finally, the ML's meta-model plays the role of the database schema. We can thus apply the database technique in order to compare MLs meta-models.

Therefore, in the UEML project, we defined a complex scenario (comprising a set of models and related model artifacts) playing the role of databases and instances, by using three distinct modelling languages i.e. IEM, EEML and GRAI. Meta-models for each of these languages were also built. Afterwards, as it happens within the database world, we were interested in comparing these meta-models based on models and model artefacts,part of the scenario. Based on such comparisons, in the simpler case of two constructs with their underlying concepts C1 and C2 respectively in EML1 and EML2, we found the following types of semantic correspondences:

1. $C1 = C2$
2. $C1 \subset C2$
3. $C1 \supseteq C2$
4. $C1 \cap C2 \neq \emptyset$

For instance, the first semantic correspondence (1) can be paraphrased as: by looking into the available models, represented in EML1 and EML2 respectively, the model artifacts represented through the concept $C1$ are also

represented by $C2$ and vice-versa.
In the general case, a common concept C can be introduced if $C1 \cap C2 \neq \emptyset$

3.2 UEML 1.0

3.2.1 Context

In 2001, a working force, called the UEML Working Group, was set up in order to contribute to the *UEML project* (www.ueml.org).

The UEML Working Group was composed of eight core members. It also contained some industrial and academic members which form a UEML network of persons interested in the enhancement of interoperable EM solutions. The UEML core group members were GRAISOFT (associated to the LAP/GRAI-University Bordeaux 1 and LABRI-University Bordeaux 1), INRIA (associated with CRAN-University Henri Poincaré Nancy 1), COMPUTAS AS, CIMOSA Association, IPK/FhG Berlin, University of Torino, University of Namur and Polyechnic University of Valencia.

The UEML project actually started on March 1st 2002 and ended May 30th 2003. It pursued the following objectives:

- to create a European consensus on a common modelling language and to facilitate interoperability within the frame of on-going standardisation efforts. The common language representing this consensus was defined in terms of a core set of modelling constructs;
- to build an UEML demonstrator to promote, test, and collect comments, validate and improve the proposed Modelling Language Constructs;
- to prepare the launching of a project to define, implement, and promote the complete UEML.

The creation of UEML 1.0 remains one of the major result of the UEML project. The objective was to define an initial core language to enable the exchange of models between three existing modelling tools (Moogo, Metis et e-Magim). These three tools supported respectively the three following enterprise modelling languages: IEM, EEML and GRAI.

3.2.2 How it works

UEML 1.0 only described the abstract syntax without taking into account any specific method for making this abstract syntax. The idea of UEML 1.0 was to state some basic correspondences between languages by using examples.
The strategy depicted in Figure 3.2 and described below was applied to define UEML 1.0 [BPP04]:

1. A scenario is defined and modeled in each language. This task has to be achieved by modelling experts of the languages. But of course, as experts are humans, a part of subjectivity is still possible.

2. Each language's meta-model is defined in a UML class Diagram. Those meta-models represent the languages abstract syntax by a set of classes and relationships between them. They are checked by UML experts.

3. Semantic correspondences are established between the three languages.

4. Common concepts and some non-common concepts are identified.

5. A version of UEML can be made with the common concepts identified in the previous step. Some non-common concepts are also added.

6. A final version of semantic correspondences between the languages and the UEML metamodel is defined.

7. A final check of the semantic correspondences between the languages and the UEML metamodel is done (for instance with new scenarios).

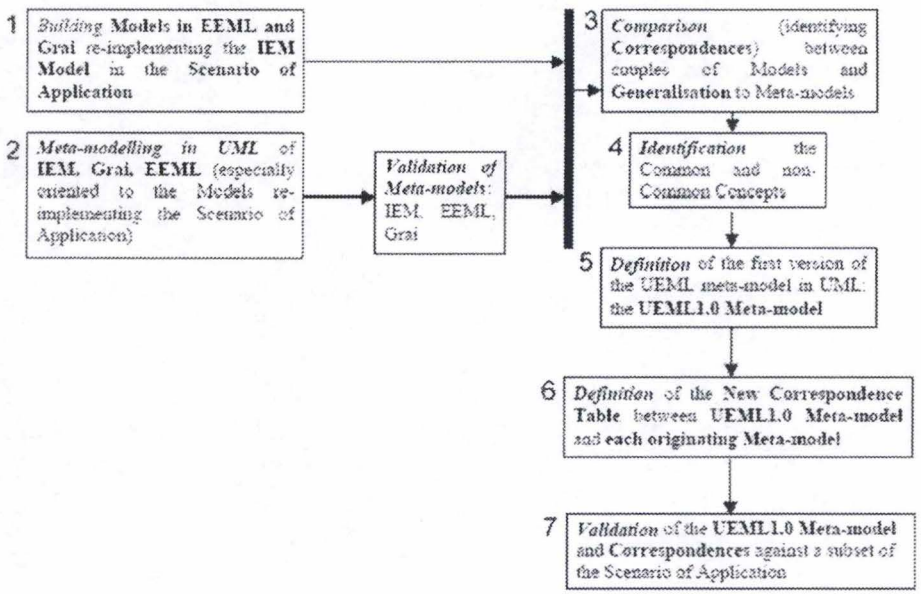

Figure 3.2: The UEML 1.0 strategy.

The application of this strategy on the three languages (GRAI, IEM, EMML) has led to produce the meta-model presented in Figure 3.3

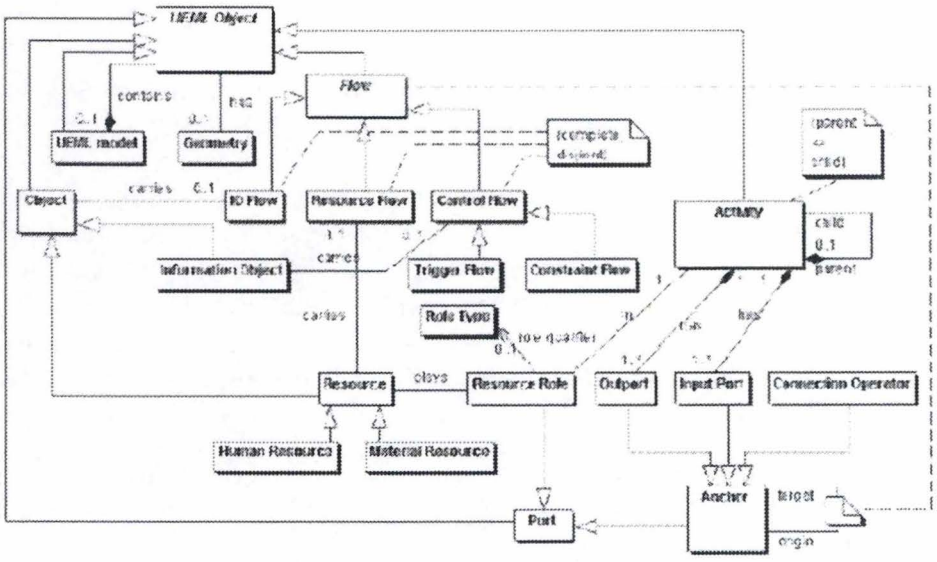

Figure 3.3: The UEML 1.0 Meta-model (from [BPP04]).

3.3 UEML 2.0

From 2003 to 2007, a second project partially supported by the Commission of the European Communities under the sixth framework program, INTEROP Network of Excellence[1], composed of 47 partners and more than 300 researchers, focused on the development of a second version of UEML, UEML 2.0.

The vision of UEML 2.0, as stated in [BOAD05] is:

"UEML brings together theories and mechanisms for "modelling enterprise modelling languages", theories and mechanisms for characterizing and finding correspondences between constructs in distinct enterprise modelling languages and, finally, strategies for selecting/classifying such languages according to users' needs."

"All these theories and mechanisms provide basic mechanisms (i.e. the correspondences between constructs in distinct modelling languages) to enable integration and integrated use (in several ways, e.g. integration, transformation, of models used in enterprise activities) of enterprise models (stored in distinct enterprise modelling tools supporting distinct languages) and, on the other hand, when applied to selected languages, to discover core concepts for

[1] (www.interop-noe.org)

enterprise modelling (represented throughout a UEML core language)."

This development of the second version of UEML included three main activities ([BO06] and [BOAD05]):

- **Requirements**: focused requirements were collected using a requirements elicitation template which is simple and based on the idea that the users should explain the needs that they submit in more detail. Specifically, the requirements template requires reformulating the needs the users are expressing in several ways and with distinct statements. Requirements are then organized into three main groups (core, basic and extended).
- **Language selection**: languages to be incorporated into UEML 2.0 were evaluated with the help of a language template and selected using a set of quality criteria.
- **Definition approach**: the incorporation of the selected languages into UELM 2.0 was made construct-by-construct using the UEML template approach.

Together, these three activities constitute the UEML 2.0 general approach.

We will now further detail the language selection and definition approach in the following sections.

3.3.1 Language selection

The concept of quality is here applied to EMLs because quality is closely related to the suitability of the language for modelling the enterprises.

The quality framework of [Ko03] has been applied and extended ("The Extended Quality Framework") to allow the selection of various languages to be analyzed and integrated into UEML.

"This framework essentially provides the neutral definition of the several quality types of a language (for instance, syntactic quality represents to what extent a language syntax corresponds to the concepts to be represented): neutral means that the quality framework can be applied to languages that are used for modelling without any regards to the specific domains. Each of these quality types is related to what is called appropriateness. The various types of appropriateness provide the context to evaluate the related quality types." [BOAD05]

1. **Domain appropriateness**
   That is used to assess physical and semantic qualities; domain appropriateness is essentially the relationship between the domain of modelling and the way of modelling using that language.

2. **Participant language knowledge appropriateness**
   The conceptual basis of the language should correspond as much as possible to the way individuals perceive reality.

3. **Knowledge externalization appropriateness**
   The goal is that there are no statements in the explicit knowledge of the participant that cannot be expressed in the language.

4. **Comprehensibility appropriateness**
   The goal is that the participants in the modelling effort using the language understand all the possible statements of the language.

5. **Technical actor interpretation appropriateness**
   For tools interpretation, it is especially important that the language lend itself to automatic reasoning.

6. **Organizational appropriateness**
   It is to what extent the language is appropriate for the organization using it, taking into account standardization on technology, tools and modelling methods within the organization.

In the UEML approach, quality criteria were defined starting from the requirements and a language template was defined for collecting "factual" information about languages. However, once several criteria have been evaluated, the selection of languages is not an easy task. Indeed, it is possible to have contradicting criteria. Therefore, in the selection activity, providing additional information about why the UEML template approach is used should be introduced in order to facilitate the selection of languages.

Figure 3.4 the two first step of the UEML 2.0 general approach:

The principles used for defining the new approach for UEML 2.0 have their foundation in the vision. They were [BOAD05]:

1. Integrative approach

2. Extendable, tailorable

3. Standardized and template-based

4. Separate presentation from content

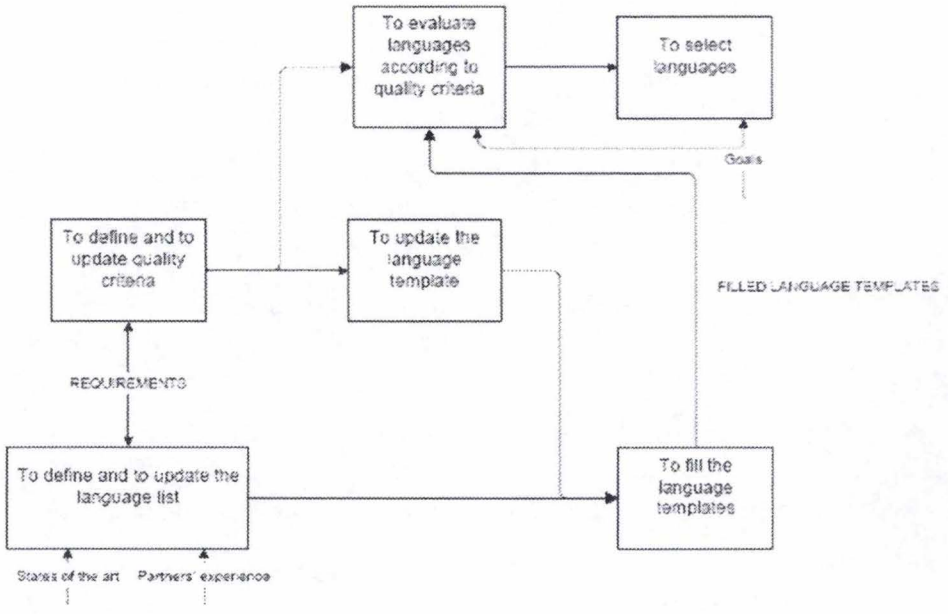

Figure 3.4: Tasks and their dependencies to perform the selection of languages for UEML.(from [BOAD05])

5. Structured approach to organize and manage the common meta-model

6. Industrial languages

To respect all these principles and model the EMLs according to their three main components (syntax (presentation), ontology (elements belonging to the domain) and semantics (representation mapping)), the "UEML template approach" has been retained. It requires a detailed (ontological) analysis of the constructs found in EMLs and allows to formally define correspondences between constructs in distinct languages and thereby a UEML-based core enterprise modelling language.

Before we further describe the UEML template approach, a clear definition of the concept of "ontology", at the core of UEML 2.0 , is provided [2]:

"In philosophy, ontology is the study of being or existence. It seeks to describe or posit the basic categories and relationships of being or existence to define entities and types of entities within its framework. Ontology can be

[2]http://en.wikipedia.org/wiki/Ontology

said to study conceptions of reality."

"In computer science, ontology describes the concepts and relationships that are important in a particular domain, providing a vocabulary for that domain as well as a computerized specification of the meaning of terms used in the vocabulary. Ontologies range from taxonomies and classifications, database schemas, to fully axiomatized theories. In recent years, ontologies have been adopted in many business and scientific communities as a way to share, reuse and process domain knowledge. In short, ontologies are the common vocabulary in which shared knowledge is represented."

Ontologies are now central to many applications such as scientific knowledge portals, information management and integration systems, electronic commerce, and semantic web services.

Let's now detail the UEML template approach. This approach is based on:

1. The **meta-meta model**, represented as a simple UML Class Diagram. It is depicted in Figure 3.5.

   It is applied for describing the abstract syntax of individual modelling constructs, an ontology that is common to all the languages (termed the common ontology) and representation mappings between them.

   It is based on a limited set of well-defined concepts (defined in Bunge's ontology and the BWW-model, see below).

   [BOAD05] *"While applied to enterprise modelling languages, the meta-meta model allows describing generically any language of a concrete domain through its constructs; therefore, the Figure 3.5 shows as a UML Class Diagram the structure of the meta-meta model, which is organized in two layers; an upper construct layer (mainly related to the abstract syntax of the construct) and a lower ontology layer (i.e. the common ontology). These two layers are related by associations that indicate what phenomena in the domain each construct is in tended to represent. Each construct has an individual abstract syntax layer, whose contents are mapped onto the common ontology layer shared by all construct definitions. In consequence, each modelling construct can be defined in detail in the private abstract syntax layer, whereas the common ontology accounts for representational overlaps (or redundancies) between different constructs."*

   The meta-meta model corresponds most directly the "Representation" section of the "UEML template" presented below.

2. A **standardization of objects** that can be used to instantiate the meta-meta model.

   Specifically, the BWW concepts (Bunges ontological model [Bun77], [Bun79] and the Bunge-Wand-Weber representation model [WW88], [WW93], [WW95] (the BWW-model), have been used for this purpose.

   These objects represent basic phenomena of the real world and are used for describing complex constructs found in the various languages. The BWW ontology is composed of [BOAD05]:

   - **Classes of things in the domain**. Classes are characterized by properties and they are organized in generalization/specialization hierarchies. A modelling construct may represent any number of such classes.
   - **Properties of things and classes in the domain**. Properties belong to things and characterize (or define) classes. Properties are organized in precedence hierarchies (being alive precedes being a mammal; being a mammal precedes being human). A modelling construct may represent any number of such properties. Properties can be mutual to two or more things or classes (i.e. "relationships"). There is also a special part-whole relation property.
   - **States of things in the domain**. States are defined in terms of properties and may be constrained by a particular kind of property, called a state law property.
   - **Events of things in the domain**. Events have a from state and a to state and may be enforced by a particular kind of property, called a transformation law property. Events may comprise other events. Such complex events, containing intermediate states, are also called processes.

   The UEML is based on Bunge's ontology and the BWW representation model. Consequently, the description of a modelling construct results from the use of ontological concepts that are maintained in a hierarchically organized common ontology. The addition of more modelling constructs to the UEML makes this ontology grow gradually.

3. A **template**, named **UEML template**.

   The need for a template arose because there were no specific computerized tools to instantiate the meta-meta model. Thus, in order to follow the approach, a text-based template was set up. Analyzing a modelling construct is done by filling it in. Some of the entries in the

template correspond to the abstract syntax layer, whereas others correspond to the common ontology. The latter entries are filled in by reusing concepts from the common ontology when possible. Furthermore, the template supports all components of modelling constructs and is, therefore, made of three parts: preamble, presentation and representation.

Here is how the template explains the entries needed for each part:

(a) **Preamble**
General issues: construct, diagram type and language names, acronyms and external resources.

(b) **Presentation**
Presentation issues: lexical information (icons, line styles), syntax and pragmatics (layout conventions).
Earlier versions of the template used the term "Syntax" to describe this section.

(c) **Representation** (also called semantics)
Semantic aspects: which instantiation level, classes, properties and kinds of dynamic behavior can the construct be used to represent. Most modelling languages only describe their semantics using text, so the entries in the semantics part of the template usually cannot be filled in with information directly from the language definition. We must look a little "between the lines", while at the same time avoiding putting our own semantics into the language.
It is likely that this part of the template will not fit well for constructs that define data types or particular values. But we are not likely to encounter many such constructs. If you do, just identify them explicitly and fill in the rest the best you can.

When analyzing new languages, the template helps to ensure that the resulting descriptions are consistent.

3.4 comparison UEML 1.0 - UEML 2.0

This section will summarize the key points of each version of UEML presented in this chapter, UEML 1.0 and UEML 2.0, by highlighting their similarities and discrepancies.

3.4.1 Similarities

Obviously, the aim of the two versions is similar: they were developed to support enterprise model exchange (integration, translation and transformation) and the required global consistency between distinct evolving enterprise modelling languages. [Ber05b]

As pointed out by [Ber05b], the approaches in both version have the same limitation: they do not allow to formally proof properties of basic correspondences and more complex exchanges.

3.4.2 Discrepancies

Concerning the approach taken to achieve their goal, the projects are different: the idea in UEML 1.0 was to state basic correspondences between languages by using specific examples. It is thus more pragmatic. The main results were related to the development of a "pivotal language" (sometimes referred to as an exchange format) that can be used to perform simple exchanges between tools supporting enterprise modelling languages. Instead, the approach adopted in UEML 2.0 allows analyzing several languages but requires the definition of a common semantic domain for languages (The UELM template approach). The approach is a more general approach.

UEML 1.0 describes the abstract syntax but does not look into the other components of the language. The UEML template approach, instead, adopted a complementary approach to UEML 1.0 by fully modelling the languages in their three conceptual components: abstract syntax, semantic domain and semantics.

As a consequence, the correspondences between languages in UEML 2.0 are not statically defined as in the UEML 1.0 approach: in UEML 2.0, they should be inferred according to the represented semantics and the semantic domain.

The UEML template approach is also complemented by a continuous collection and elicitation of requirements. Based on these requirements, clear strategies are defined for selection/classification of existing languages as relevant for UEML. Such selection and classification of existing enterprise modelling languages is clearly a key pre-requisite for the development of a useful and used UEML [BOAD05] and is absent from UEML 1.0.

Finally, UEML 2.0 guides, according to the met-meta model, the representation of abstract syntax, semantics and semantic domain whereas UEML 1.0 does not take into account any specific method for making the abstract

syntax.

3.5 Summary

During this chapter, we found out the purpose of UEML. We gave a general overview of the first version of UEML, UEML 1.0. Then, we described much more in details the second version of UEML, UEML 2.0. We set out UEML 2.0 main activities and we put a particular emphasis on the UEML 2.0 approach which is used to perform compliant language analysis. Finally, we provided a comparison between UEML 1.0 and UEML 2.0.

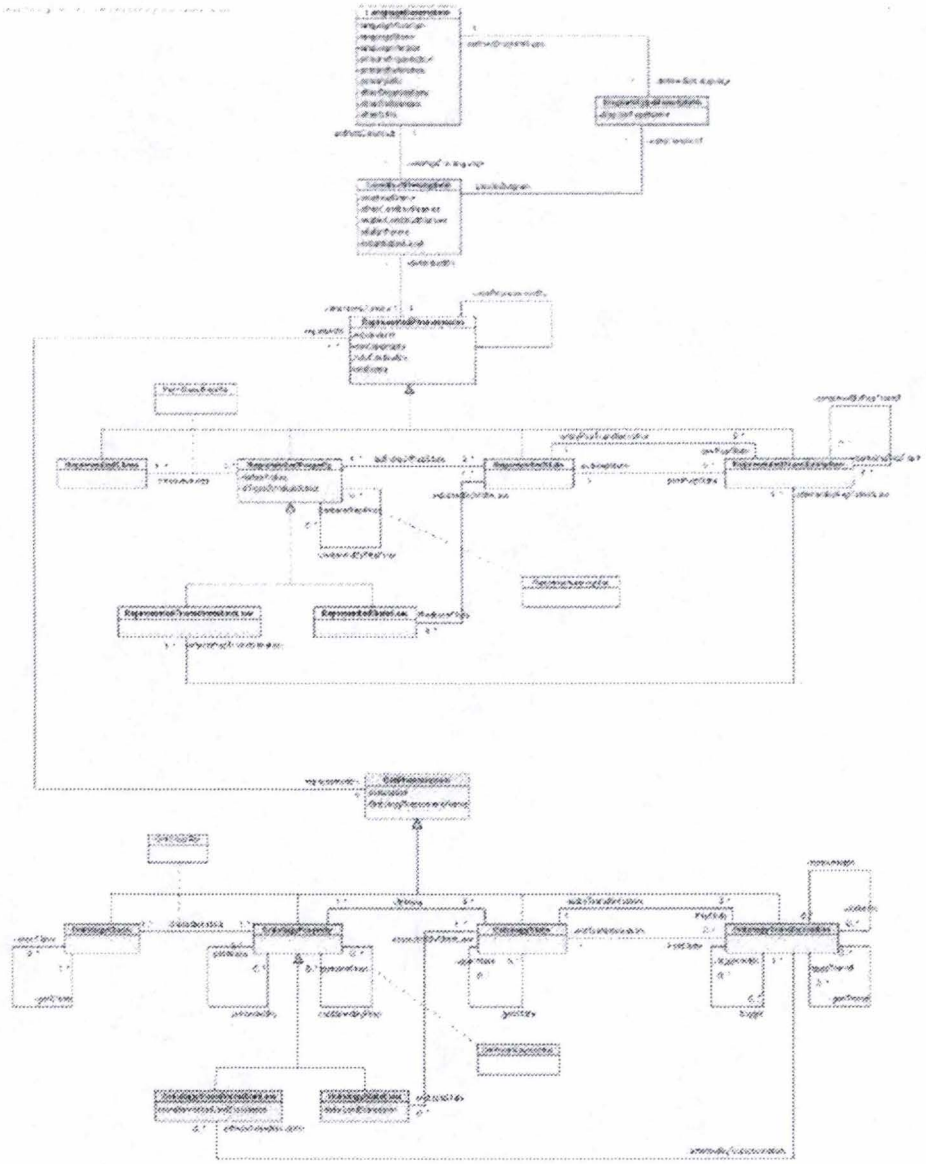

Figure 3.5: The UEML 2.0 meta-meta model.(from [BO06])

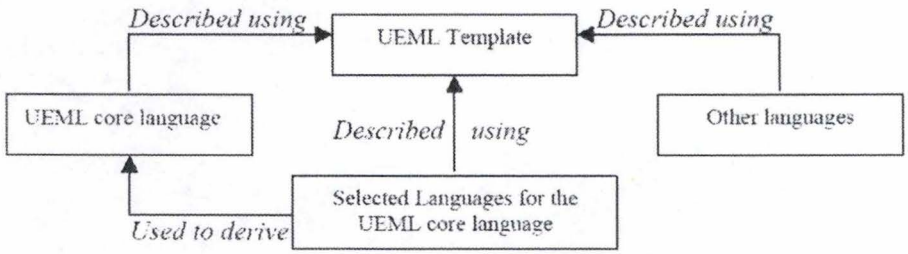

Figure 3.6: The UEML template approach allows to include languages other the selected languages for developing a UEML core language.(from [BOAD05])

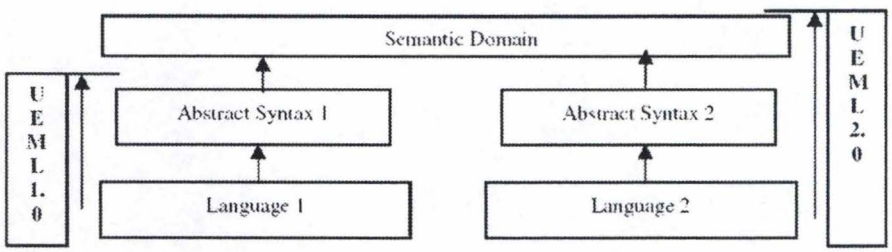

Figure 3.7: comparing UEML 2.0 and UEML 1.0 approach (from [Ber05b])

Chapter 4

UEML tools

This chapter aims to provide a description of the different UEML tools that will be used during the contribution and the evaluation part of the document. First, it will present Protégé, a tool which was used to perform language analysis. Then, it will set out the UEML Validator which was used to validate the language analysis performed. Finally, it will present a Protégé Plug-In, called similarity Plug-In, which is used to evaluate the similarity between constructs and which was tested out on three different EMLs.

4.1 Protégé

This section will describe Protégé, a open-source platform which was used to perform language analysis[1]. In the context of EMLs analysis, Protégé aims to provide a support to the UEML 2.0 approach by providing solutions in order facilitate the language analysis.

4.1.1 About Protégé

Protégé is a free, open-source platform that provides a set of tools to a growing users community to construct domain models and knowledge-bases applications with ontologies. Implementing a rich set of knowledge-modelling structures and actions at its core, Protégé enables the creation, visualization and manipulation of ontologies in various representation format. By customizing Protégé, domain-friendly support for creating knowledge models and entering data can be obtained. Futhermore, building knowledge-based tools and applications is made possible with an extension of Protégé by way of a plug-in architecture and a Java-based Application Programming Interface (API). Two main ways of modelling ontologies are supported by Protégé platform:

[1]This section is partly inspired by information found on http://protege.stanford.edu/

- Users can employ Protégé-Frames editor to build and enrich ontologies that are framebased, in accordance with the Open Knowledge Base Connectivity protocol (OKBC). In the framework of this model, an ontology is composed of:

  1. a set of classes organized in a subsumption hierarchy to represent a domain's salient concept;
  2. a set of slots associated to classes to describe their properties and relationships;
  3. and a set of instances of those classes, individual exemplars of the concepts that hold specific values for their properties.

- By mean of Protégé-OWL editor, which is an extension of Protégé that supports the Web Ontology Language (OWL), users can build ontologies for the SemanticWeb, in particular in the W3C's Web Ontology Language. An OWL ontology may include descriptions of classes, properties and their instances. Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics. These entailments may be based on a single document or multiple distributed documents that have been combined using defined OWL mechanisms.

  The Protégé-OWL editor enables users to:

  - Load and save OWL and RDF ontologies.
  - Edit and visualize classes, properties and SWRL rules.
  - Define logical class characteristics as OWL expressions.
  - Execute reasoners such as description logic classifiers.
  - Edit OWL individuals for Semantic Web markup.

  The configuration and the extension of the tool is easier with Protégé-OWL flexible architecture. Protégé-OWL is tightly integrated with Jena (a Java API for OWL) and has an open-source Java API for the development of custom made user interface components or arbitrary Semantic Web services.

  A user of the Protégé-OWL editor does not actually need to know the OWL syntax in order to manage an ontology. It is done logically. The graphical interfaces fundamentally simplifies the creation of classes and properties. The form from the "class" or "property" tab provided by Protégé interfaces merely have to be filled in. To update those classes and properties can be done by simply changing the values of the different fields. An ontology can be easily populated by creating adapted

forms.
Individuals are created with the same process as for classes and properties, thanks to Protégé graphical interfaces. The individuals can be visualized by means of the "individuals" tabs. Their properties can be changed by using the "individuals" tabs.

Within the context of UEML, a major contribution brought by Protégé is to have a unique base of knowledge and to formalize the analyses. This knowledge base, coded in an OWL file, is called the UEMLBase.

4.2 A validation Plug-In for UEMLBase

4.2.1 Overview

The UEML Validator is a tool that allows to check the UEMLBase consistency. In this regard, it verifies if a set of pre-defined constraints are respected in the whole OWL file containing the analysis results.
The results of the validations performed by the tool allow the language analyst to improve its language analysis.

4.2.2 How it works

This section is inspired by [Mah06].

The UEML Validator has a pipe and filter architecture. Indeed, as we can see in Figure 4.1 it takes the informations of the OWL file in order to make a Prolog fact base. Then it adds the rules written in a separate file and finally, it checks them with the help of another file and shows the errors. The UEML Validator has been made as independent from the model and the constraints as possible. The model of which we want to check constraints is not important. The only thing it has to respect is to have been written in OWL. Indeed, it generates the prolog base without knowing the name of the classes. The model has thus just to be written in OWL and to respect its syntax. This is the same for the rules that have to be checked. But even so, they have to respect two constraints:

1. to be written in Prolog;
2. to correspond to the way facts are generated.

The model and the rules are not hard coded in the program so they have to be given to the program.

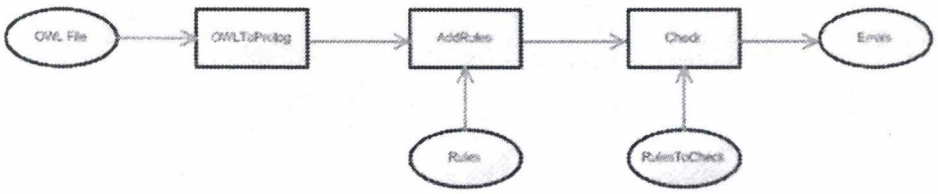

Figure 4.1: UEML Validator's pipes and filters architecture. (from [Mah06])

4.3 A similarity Plug-In for UEMLBase

4.3.1 Overview

The similarity Plug-In is a Protégé tool which allows to perform some similarity comparisons between constructs contained in a common ontology. When the tool computes a similarity comparison between two selected constructs, it produces a result contained between 0 and 1. This result must be interpreted in regard to the measure form selected. The next section explains in details how the similarity Plug-In computes its results and how to interpret them.

4.3.2 How it works

In order to work appropriately with regard to the context, the similarity Plug-In offers the possibility to configure a set of four parameters[2]:

1. The first parameter that can be configured is the **Root Status**.

   The root status determines if the ontology root will be used to bring an additional information to the similarity between the compared constructs (informative root) or not (virtual root). If the informative root is selected, it means that if two described constructs have a same ontology root, it will increase their similarity, so the tool will consider a common root as a relevant similarity information. Conversely, if the virtual root is selected, a same ontology root between two constructs compared won't be taken into account by the tool.

2. The second parameter that can be configured is the **Distribution Hypothesis**.

   The plug-in allows the user to choose between four distribution hypothesis. H1 (uniform on depth level), H2 (uniform on sons), H3 (uniforms

[2]This section is based on explanations provided by Mounira HARZALLAH - University of Nantes

on leaves) and H4 (arithmetic mean). These distribution hypothesis aims to measure the repartitions of the instances of the ontology concepts. Therefore, it is relevant to work with the same hypothesis all along the comparison phase. The different hypothesis work as follows:

- H1: the number of instances of ontology concepts is devided by a scalar for with each specialization;
- H2: the distribution of instances of ontology concepts is uniform on all the sons of each concept;
- H3: the distribution of instances of ontology concepts is uniform on all the leafs of the taxonomy;
- H4: takes into account the last two hypothesis.

3. The third parameter that can be configured is the **Measure Form**.

   Once the distribution hypothesis defined, one must choose the kind of measure form.
   The measure forms are based on the fact that the similarity between two constructs depends on their commonalities and on their description. What describes a construct can be defined by its informational content and what they have in common can be defined by the ontological concepts that they subsume. The informational content of a construct C is defined in relation to the probability that a given ontological instance could be an instance of C.

   Three kind of measure forms are available: Jaccard, Precision and Recall. The result of a measure form will depend on the commonalities between two constructs and what differenciate them.

   How to interpret a result depends actually on the measure form chosen. Jaccard is characterized by a strict similarity of the compared concepts if the result is 1. Precision or Recall have a slightly different approach. They work in terms of inclusion of one set in another (typically some constructs) if the result is 1.

   Let f be a function which evaluates the quantity of information of the constructs and *C1*, *C2* two constructs. f depends on the repartion of instances. The three Measure form are defined as follows:

   - Jaccard = $f(C1 \cap C2)/f(C1 \cup C2)$
   - Precision = $f(C1 \cap C2)/f(C1)$
   - Recall = $f(C1 \cap C2)/f(C2)$

4. The fourth parameter consists to adjust the **Importance Coefficients** one wants to give to the classes, the properties, the states and the transformations during the computation of the similarity.

The similarity Plug-In interface and its set of four parameters are depicted in Figure 4.2

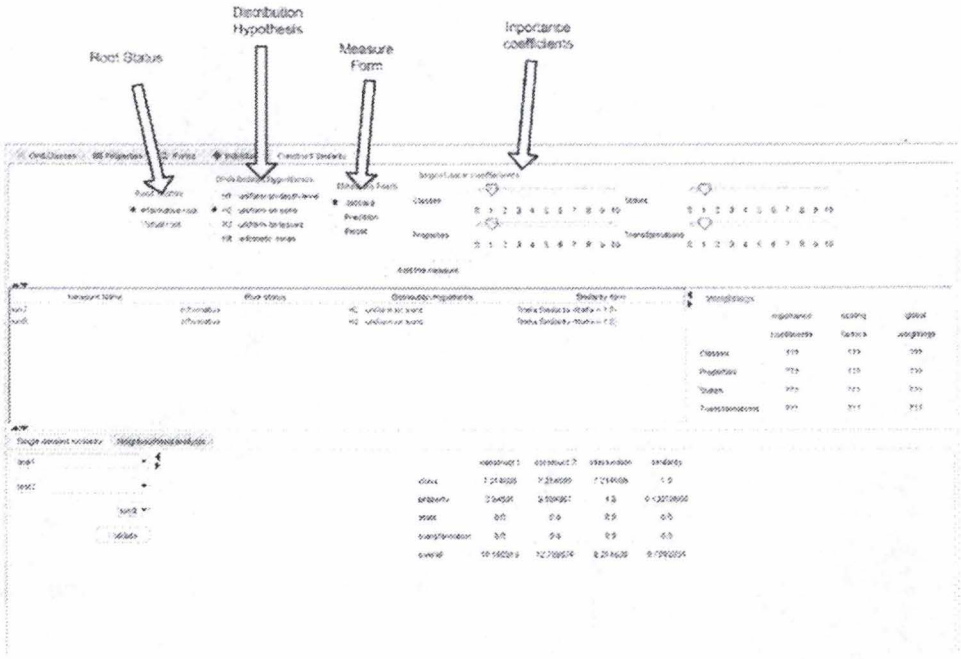

Figure 4.2: The similarity Plug-In interface and its different parameters

4.4 Summary

This chapter provided an overview of the major tools used throughout this work. We first presented Protégé and more specifically its OWL editor which allows users to build ontologies. Then, we introduced the UEML Validator which purpose is to check the UEMLBase consistency and therefore to allow the analyst to improve his language analysis. Finally, the last section of the chapter presented the similarity Plug-In which allows to perform some similarity comparisons between constructs contained in a common ontology.

Chapter 5

Three Enterprise Modelling Languages

This chapter will introduce three EMLs: BMM, *i** and GRL. BMM and *i** were the subject of language analysis during this work whereas GRL will serve as a basis for testing out the similarity Plug-In presented in section 4.3.

5.1 BMM

5.1.1 History of the Business Motivation Model

The Business Motivation Model (BMM) was first published by the BRG (Business Rules Group) in 2000, under the title "Organizing Business Plans The Business Rules Motivation Model". It was updated in 2005 with a new title "The Business Motivation Model - Business Governance in a Volatile World".

In September 2005, BMM was accepted by the Object Management Group (OMG) as the subject of a Request for Comment. This means that the OMG is considering the BMM as a de-facto standard that could be adopted as an OMG specification, subject to comment by the industry.

In 2006, BMM became an official OMG specification.

5.1.2 Presentation of the Business Motivation Model

This section is inspired by [OMG06].

The Business Motivation Model offers a scheme or structure by way of which business plans can be developed, communicated and managed in an organized manner. In particular, the Business Motivation Model achieves the following:

- Identification of factors that motivate the establishing of business plans.
- Identification and definition of the elements of business plans.
- Indication on how all these factors and elements inter-relate.

The business Motivation Model is depicted in Figure 5.1. We can identify two major areas in the Business Motivation Model:

- the first is the Ends and Means of business plans. Ends correspond to the objectives that the enterprise wishes to reach - for example, Goals and Objectives. Means are the tools that the enterprise will use to achieve those Ends - for example, Strategies, tactics, Business Policies and Business Rules.
- the second is the Influences that determine the elements of the business plans and consequently, the Assessments given about the impact of such Influencers on Ends and Means (i.e., Strengths, Weaknesses, Opportunities, and Threats). The Ends, Means and Influencers are interrelated to give an answer to the following two fundamental questions:

  1. what is needed to allow the enterprise to realize its goals?
     To answer that question requires outlining the specific elements of the business plans or, in other words, the Means necessary to achieve the desired Ends.
  2. what justifies the presence of each element of the business plan?
     The identification of the particular Ends that each Means serves and the Influencers that underlie the choices made in this respect give an answer to this question. That is the meaning of motivation.

The development of all the elements of the Business Motivation Model is business oriented. Basically, the idea is to develop a business model for the elements of the business plans prior to the system design or the technical development. In this way, the business plans become the starting point for such activity by connecting tightly system solutions to their business purpose.

5.2 *i**

5.2.1 Context

The *i** has been designed for modelling organizations as a reasoning tool to apprehend changes in relationships among strategic actors. It's agent-oriented because a number of research areas is focusing on this approach,

including information systems requirements engineering. i^* considers organizations as social actors who can act freely but are depending on each other to achieve goals, to perform tasks and to furnish resources. The framework comprises a Strategic Dependency model for the description of the network of relationships among actors and a Strategic Rationale model for the description and the backup of the reasoning that each actor has about its relationships toward other actors. These relationships are strategic given that each actor is concerned with opportunities and vulnerabilities and is seeking to protect or promote its interests.

5.2.2 The Strategic Dependency model

[Yu95] describes the Strategic Dependency model as follows:

"The Strategic Dependency (SD) model is a network of dependency relationships among actors. The intuitive meaning of a dependency is that a depender by depending on someone else (the dependee) for something (the dependum) can accomplish some goal or objective that it would otherwise be unable to achieve (or not as well). If the dependum is not forthcoming from the dependee, the depender would suffer as a result, i.e., its attempt to accomplish the objective may fail or may be compromised.
The SD model therefore aims to capture the intentional structure of a process, instead of the usual non-intentional, and non-strategic process models of activities and entities. It is a higher level characterization of a process because it captures what matters to the actors, while leaving out non-essential details. The model distinguishes among several types of dependencies based on how agents constrain each others freedoms, and the extent to which they are vulnerable in their dependencies. Dependencies are threaded through roles and positions, as well as physical agents, creating an intricat web of relationships."

5.2.3 The Strategic Rationale model

[Yu95] defines the Strategic Rationale model as follows:

In the Strategic Rationale (SR) model, the rationales behind process configurations can be explicitly described, in terms of process elements and relationships among them. The main types of relationships are represented as means-ends links and task-decomposition links. Means-ends links are seen as applications of generic rules in particular contexts. Process elements include subgoals, subtasks, resources, and softgoals. The model is strategic in that elements are included only if they are considered important enough to affect the achievement of some goal. Agents may be able to accomplish something by themselves, or by depending on other agents An interconnected collection

of process elements serving some purpose for an agent is called a routine. An agent often has more than one routine for accomplishing something. Process reengineering involves modelling existing routines (e.g. by asking "why" and "how" questions) and discovering new and better routines.

5.3 GRL

Because it comes from the IS and requirements engineering communities, GRL is a EML but is not brought forward as such. However, it's a great help in EM and especially in goal-oriented modelling and reasoning about requirements, in particular when it comes to non-functional requirements. Detailed specification of what is to be done differs from this kind of modelling. In the present case, the modeler's primary concern is to expose "why" certain choices for behaviour and/or structure were made or constraints introduced.
GRL supplies constructs needed to express the various types of concepts that occur during the requirements process. It's composed of four main categories of concepts : *intentional elements*, *links*, *actors* and *non-intentional elements*.

5.3.1 Intentional elements

The intentional elements in GRL are made up of *goal*, *softgoal*, *task*, *resource* and *belief*. They are called intentional because they are incorporated into models that answer to questions related to:

- the choice of particular behaviors, informational and structural aspects to be included in the system requirements;
- the alternatives that were considered;
- the selection criteria faced with alternative options; and
- the reasons to favour one alternative rather than the other.

A *goal* is a condition or state of affairs that the stakeholders would like to achieve. As a *goal*, a *softgoal* is a condition that the stakeholder wants to achieve, but there are no clear-cut criteria to determine whether this condition is achieved or not (e.g., Quick, Low effort). A *task* describes a particular way of doing something (e.g., Organise meeting or Schedule meeting). A *resource* is an entity for which the main concern is whether it is available (e.g., Proposed dates). A *belief* is used to express design rationale.

5.3.2 Links

Means-ends, *decomposition*, *contribution*, *correlation* and *dependency* fall within the category Link. How goals are practically achieved, especially through tasks is described by using the *Means-ends*. An alternative means for achieving the goal is provided by each task. Normally, each task would impact the softgoals differently which, in turn, would serve as assessment and selection critera for each task alternative. The *decomposition* gives a definition of the subcomponents of a task, especially (but without beeing restrictive to) the subgoals that must be achieved. The *contribution link* shows how, by design, on element impacts another one (i.e., how softgoals, task, believes or links contribute to others). A contribution is an effect that is a primary desire during modelling. Expressing knowledge about interactions between intentional elements in different categories, the *correlation* allows to encode such knowledge. A correlation link is similar to a contribution link except that it's not an explicit desire but a side-effect. The *Dependency*link gives a description of an intentional relationship between two actors (i.e., one actor (Depender) depends on another actor (Dependee) for something (Dependum)).

5.3.3 Actors

Actors are holders of intentions and characterize active entities (e.g., Meeting initiator, Meeting participant). They want goals to be achieved, tasks to be performed, resources to be available and softgoal to be "satisfied". Obviously, an actor may optionally have a boundary, with intentional elements inside.

5.4 Summary

This section introduced BMM, i^* and GRL. It showed the specificities of each language and the modelling approach that they respectively undertake. The next part of the document will present an ontological analysis of BMM and i^* in accordance with the UEML approach.

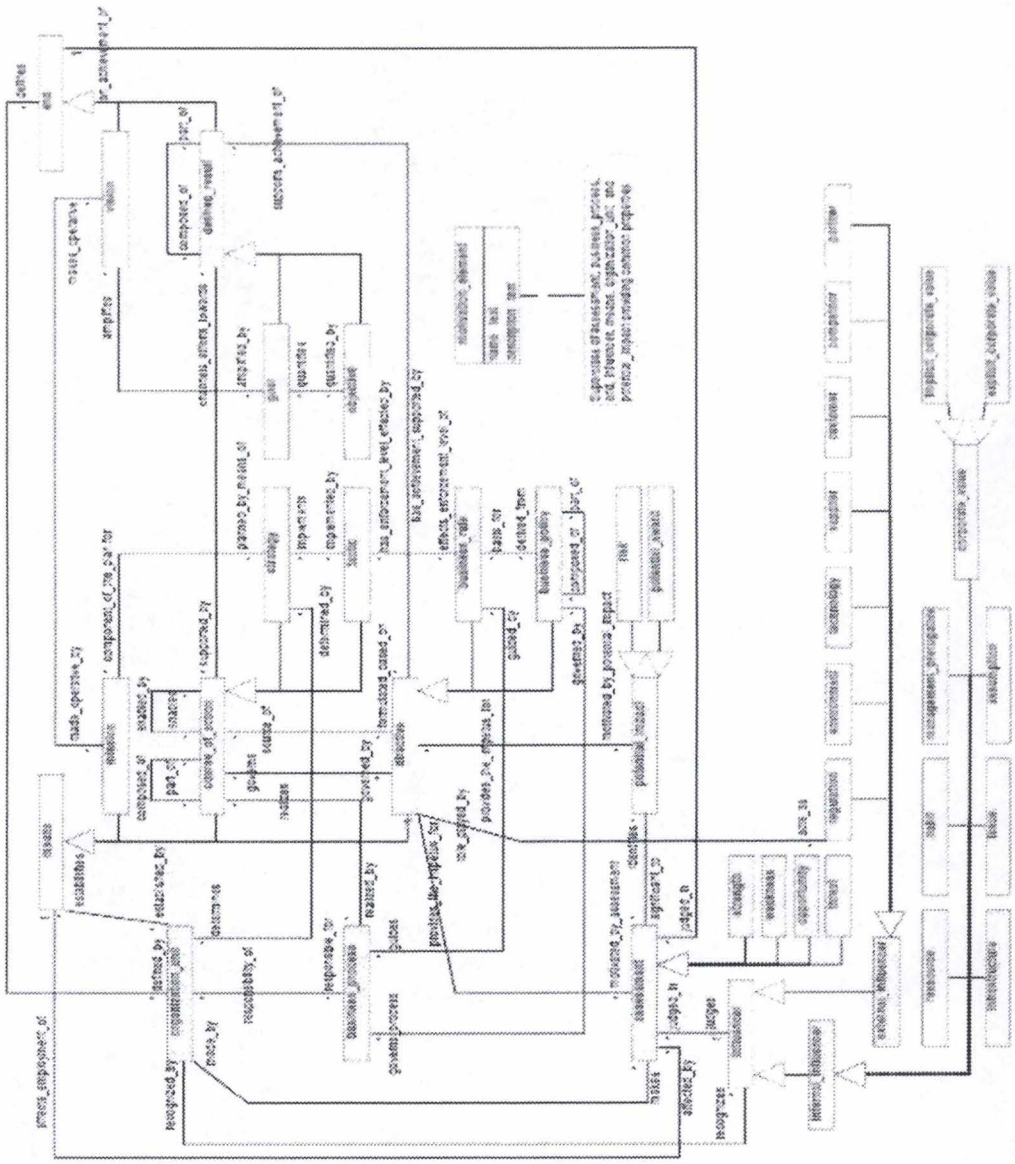

Figure 5.1: The Business Motivation Model.(from [OMG06])

Part II

Contribution

Chapter 6

Language analysis methodologies

The analysis of a language in accordance with the UEML approach is a complex process. The language analyzt must go through several important phases when he uses the UEML approach to analyze an EML. One of them is a constructs description phase which consists of providing, for each construct of the language, a description of what it represents in the real world. Another phase is a constructs mapping phase where each construct of the language is mapped onto a common ontology. Actually, the UEML text-based template is a well established structure to follow in order to perform these phases but sometimes it is not very intuitive for the layman in the field. Therefore, this chapter aims to build up and to define two methodologies in order to support the use of the template. They will be put into practice later in chapter 7 and 8.

6.1 Construct description methodology

The description methodology proposed is modelled and represented in Figure 6.1.
It basically consists of an iterative process that the language analyzt must repeat until he considers that the construct description he is processing is cleared up of any ambiguities.
Here are precisely the steps that a language analyzt will have to go through and iterate while using this methodology:

- During the first step, the analyzt must detect any information that he considers ambiguous (that can appear as a lack of precision for instance) in the construct description given in the official language specification. If the analyzt finds any ambiguities in the description, he must consider the second step. If not, the current construct description

is kept and the iterative process is over;

- the second step of the iterative process consists of remastering a new construct description thanks to the official specification of the language and to other relevant sources of information that the analyzt chooses to use. Those new sources of information will vary depending on the judgement of the analyzt and on the language analyzed. It can be information taken from documents referenced in the official specification of the language or some information extracted from a similar language specification for instance. Once a new construct description is obtained, it must be reappraised and compared with the initial construct description in the first step.

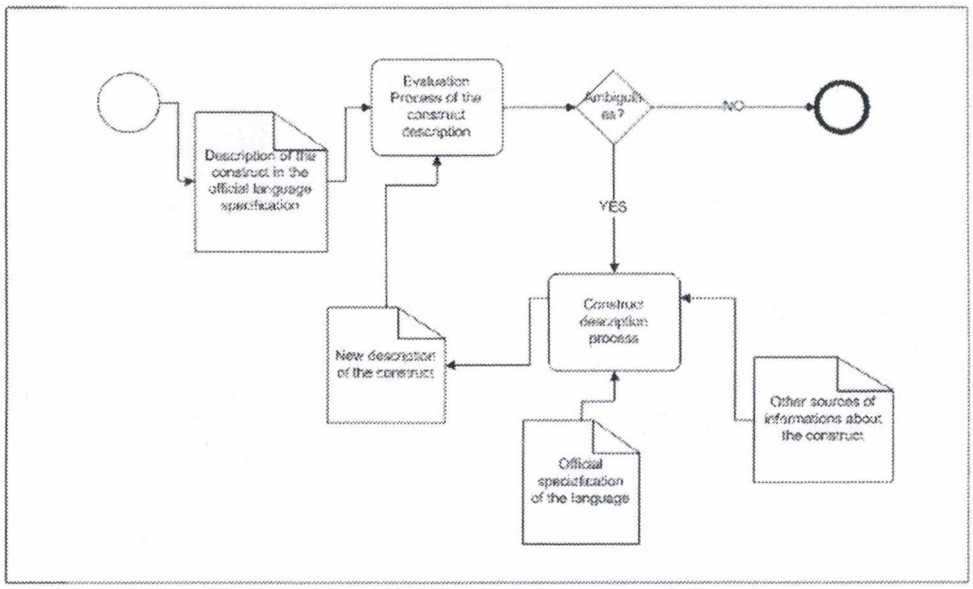

Figure 6.1: The model representing the methodology for describing constructs.

6.2 Constructs mapping methodology

There are two steps the analyzt must go through in a construct mapping process:

- Step 1 consists of breaking up the construct description into several information units and of gathering the most interesting information units in order to form a bunch of selected characteristics that will be used during the mapping operation described in step 2. This technique is presented in Figure 6.2.

The technique takes the construct description as an input and works as follows:

1. The first operation consists of finding out the *"Who"* and the possible details concerning the *"Who"* in the construct description. The *"Who"* can be defined as the answer to the question: **"What (or Who) does the description aim to describe?"**

2. The second operation consists of finding out the *"What"* and the possible details concerning the *"What"* in the construct description. The *"What"* can be defined as the answer to the question: **"What is (are) the primary purpose(s) of the *"who"* ?"**

3. The third operation consists of finding out the *"How"* and the possible details concerning the *"How"* in the construct description. The *"How"* can be defined as the answer to the question: **"How does the *"Who"* reaches its primary purpose(s)?"**

4. The fourth operation consists of selecting and gathering the "most interesting" information units on the basis of the outputs of the three first operations. "Most interesting" is a subjective notion and the choice of the selected information is left to the analyzt's judgement. The criterion he must take into account to perform the selection is the interest that a given information represents for the next step, the construct mapping itself. This selection of information units constitutes a bunch of characteristics of the construct and will be used for the construct mapping in step 2.

There is no obligation for the second and the third operation to provide some results. However, the first operation must lead to find out the *"Who"* and its details otherwise the construct description wouldn't make sense and wouldn't be valid. Moreover, in any case the *"Who"* must be kept after the fourth operation. Indeed, the *"Who"* is always the core concept of a construct description and of a construct mapping.

- Step 2 consists of performing a mapping of the construct characteristics selected in step 1 into the common ontology. For each characteristic, one (or many) instance of an ontology concept is created in order to represent the phenomenon symbolized by the characteristic. This operation requires a good knowledge of the concepts defined in the ontology and a good common sense. There is no established technique to perform a good mapping. The analyzt performs the construct mapping that he considers to be the best.

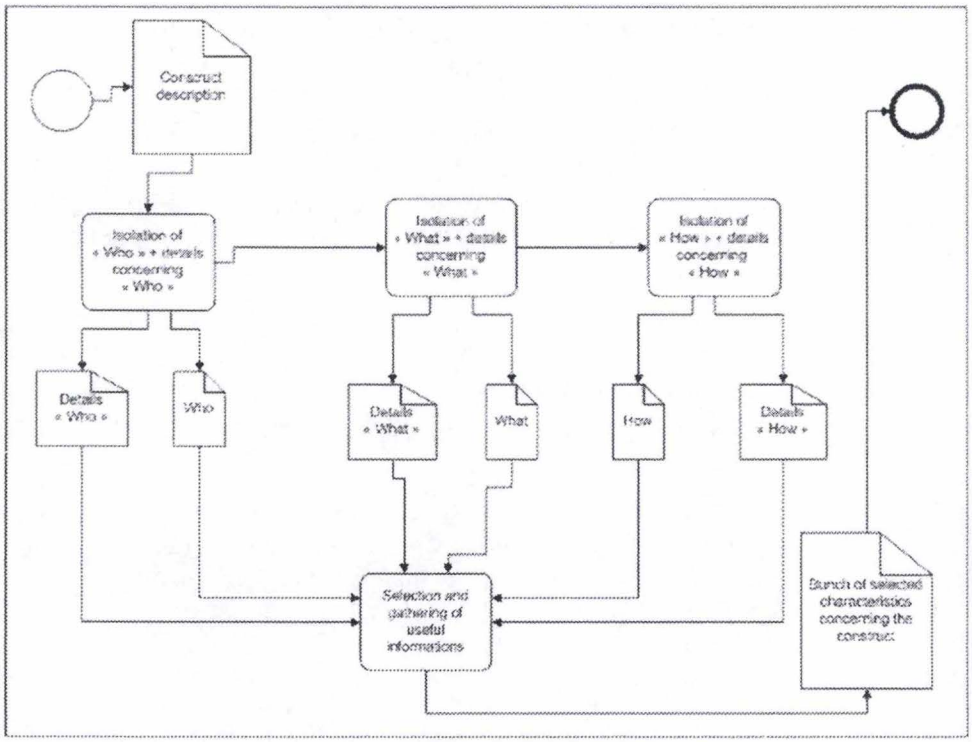

Figure 6.2: Model presenting the technique to select and to gather certain information units of a construct description.

6.3 Summary

This chapter presented two methodologies, one to perform some constructs descriptions and the other one to realize some constructs mappings into a common ontology. Those methodologies will be useful in chapter 7 and 8 where we will analyze two EML's, respectively BMM and *i**.

Chapter 7

BMM analysis

This chapter aims to present the BMM analysis. It is divided into three parts. The first part will show how the BMM constructs are described through an example. The second part will emphasize a specific case of description problem encountered with the BMM Meta-Model. Finally, the third phase will demonstrate how the BMM constructs are mapped into the UEML ontology through the same construct example discussed in the first part. The other BMM construct mappings are given in Appendix.

7.1 BMM constructs descriptions

Because of the numerous BMM constructs, we won't consider all of them in the present section but rather choose one of them and see how the construct description methodology (see section 6.1) was applied to it.

We need a criterion to select this construct. The criterion we will take into account is based on the ambiguities found in the construct description of the BMM official specification ([OMG06]). Indeed, to serve as an example, we will select a BMM construct whose official description presents some ambiguities. In this way, we will see how to clear up those ambiguities thanks to the methodology presented in Figure 6.1.

The selected construct is "Organization Unit".

7.1.1 Step 1: Evaluation process

[OMG06] describes the *Organization Unit* construct as follows:

"Organization Unit: Organization that is part of another organization (with an 'organization' being "a named group of people with a purpose and a budget")".

This description presents a double ambiguity we will have to clarify:

- The first aspect of the ambiguity comes from the information contained in the description. Indeed, the statement *"Organization that is part of another organization"* raises two questions:
  1. Does it imply that if an organization is not part of another, it can not be considered as an Organization Unit?
  2. Is this precision useful or relevant to understand the construct itself and for BMM in general?
- The second aspect of the ambiguity comes from the lack of information concerning several aspects of the Organization Unit:
  1. Is the purpose of the Organization Unit imposed by an authority or is it set by the Organization Unit itself?
  2. Is the Organization Unit involved in the processes to reach this purpose and does it decide how to reach it?

There are some ambiguities that need to be solved in the description. Therefore, the current description cannot be kept. Thus we must go to step 2 in order to remaster the initial description.

7.1.2 Step 2: Description process

In this section, we will clarify one by one the four ambiguities emphasized in step 1 and we will remaster a new description of the Organization Unit construct.

The first source of information that we will consider is the BMM specification itself ([OMG06]). Indeed, it contains some extra information about the construct that could be useful to clarify the ambiguities. The document reveals the following precisions concerning the Organization Unit ([OMG06]):

"Three concepts (Organization Unit, Business Process and Business Rule) have roles in the structure of the Business Motivation Model but actually belong in other standards, where they are defined and associated with related concepts needed for detailed business modelling."

"In the BMM as published by the BRG, the roles of Organization Unit and Business Process are described in the appendix on the Zachman framework."

Now, we know that we have some chances to find some interesting information in the BRG specification of BMM ([BRG05]). Indeed, the Appendix of [BRG05] states:

"The Model uses the following definition for "organization" on a provisional basis: "Any named group of people within the enterprise with a purpose and a budget." An Organization Unit is simply an organization that is part of another organization. These definitions should be considered merely placeholders. As stated in the main body of the document, "Organization Unit" does not fall within the scope of the Model."*

Concerning *"placeholders"*, [BRG05] precises:

"(*)In 2004, the OMG issued an RFP for an Organization Structure Metamodel, from which a standard is expected to emerge by 2006. The BRG anticipates that these placeholders will provide a link to this or other appropriate standards."

Now, we know that "Organization unit" has a role in the structure of BMM, but that it belongs to another standard. [BRG05] reveals that the concept refers to OSM (Organization Structure Metamodel), a hypothetical OMG standard which was expected for 2006. Nowadays, OSM is still under process to become an OMG standard [1]. However [DES05] proposes a core OSM Meta-Model and presents its different parts. We are interested in one specific part of the Meta-Model: the one concerning the Organization Unit. It is represented in Figure 7.1:

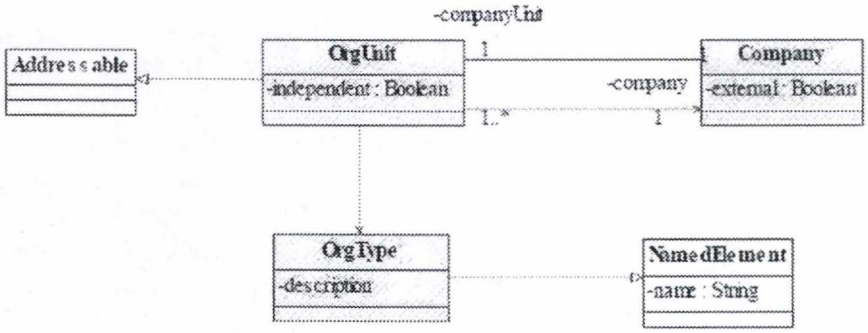

Figure 7.1: The part of the OMS Meta-Model concerning Organization Unit ([DES05]).

[1] *http://www.column2.com/category/omg/*

The Figure 7.1 allows us to clarify the first ambiguity: Does it imply that if an organization is not part of another, it can not be considered as an Organization Unit? The answer is no. The Figure 7.1 brings the answer to this specific case thanks to the cardinalities between *"OrgUnit"* and *"Company"*. We find out that:

- The first link (companyUnit) represents the situation where a whole company is the Organization Unit itself. In that case, one *"Company"* is composed of only one *"OrgUnit"* and one *"OrgUnit"* is part of only one *"Company"*;
- the second link (company) represents the other possible situation. One specific *"OrgUnit"* belongs to only one *"Company"* and one *"Company"* is composed of at least one *"OrgUnit"*.

Now, we are also able to clarify the second ambiguity: is this precision useful or relevant to understand the construct itself and for BMM in general? The answer is yes. We found out that an Organization Unit could be part of a company as well as representing the company itself. As both situations are valid, it is useful and relevant to precise that an Organization Unit is *"an Organization that is part of another organization"*. In fact, this precision allows the BMM Organization Unit to be distinguished from a classic Organization. Indeed, if we consult [OXF] and [MWU], two sources of information frequently used by [OMG06] to set up its construct descriptions, they define "Organization" as follows:

Organization: "a group of people who form a business, club, etc together in order to achieve a particular aim ([OXF])."

Organization: "an administrative and functional structure (as a business or a political party); also: the personnel of such a structure ([MWU])."

None of these definitions brings the precision about the inclusion of an organization in another. This is the reason why we consider this precision usefull and relevant within the context of BMM Organization Unit.

Besides, the Figure 7.1 emphasizes other useful information concerning the constructs. The Meta-Model shows that an organization Unit has a certain *"Type"* which is characterized by a *"description"*.

Now that we have clarified the two first ambiguities, we will focus on the lack of precision encountered in step 1 and described by the third and the fourth ambiguities.

[OMG06] provides a part of the BMM Meta-Model focusing on Organization Unit and its relations with other concepts (Figure 7.2) and which is really interesting in regard to the ambiguities we still have to solve.

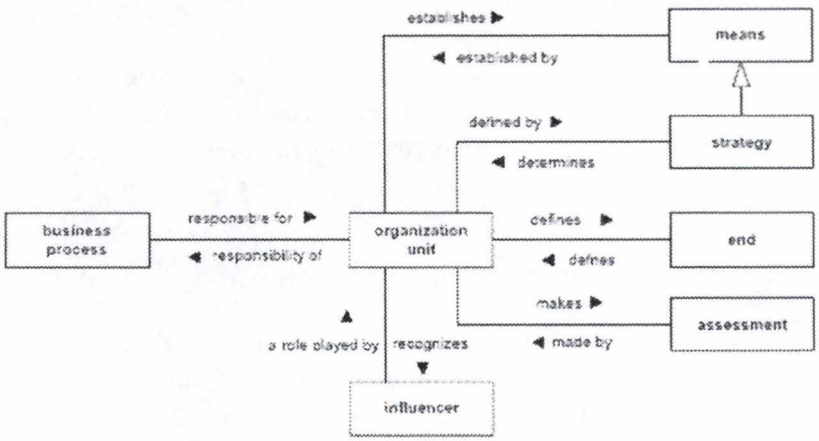

Figure 7.2: The part of the BMM Meta-Model concerning Organization Unit ([OMG06]).

Indeed, the Model represented in Figure 7.2 brings answers to the third and the fourth ambiguities. Here are the key relations that allow to answer simultaneously to "Is the purpose of the Organization Unit imposed by an authority or is it set by the Organization Unit itself?" and "Is the Organization Unit involved in the processes to reach this purpose and does it decide how to reach it?":

- *"Organization Unit defines End"*[2];
- *"Organization Unit establishes Means"*;
- *"Organization Unit is responsible for Business Process"*.

In other words, the Organization Unit is responsible for:

- defining the purpose that it wants to reach;
- defining the means to reach the purpose;

[2]The Figure 7.2 taken from [OMG06] contains several errors in comparison with the complete official BMM Meta-Model presented also in [OMG06]. In the relation between Organization Unit and Business Process, the names of the associations are inverted. In the relation between Organization Unit and End, the association that goes from End to Organization Unit is not *"defines"* but *"defined by"*.

- defining the business processes to implement the means.

Now that we have clarified all the ambiguities of the initial description, we will remaster a new description thanks to the initial description and also to the several new information that we have emphasized during the description process. Here is the remastered and final construct description:

An Organization Unit is a named group of people, eventually part of a bigger group of people, and described by various characteristics. It defines its own purposes, establishes which means to apply to reach its purposes within the limits of its budget and decides how to implement those means.

7.2 Problem encountered with the BMM Meta-Model

The ambiguities that we had to resolve in the previous section occured during the evaluation process (step 1) of the construct description methodology (see Figure 6.1). The second and the third step of the methodology were consequently applied to clarify the initial ambiguities.

This section aims to underline that some ambiguities can also arise in the third step of the methodology (precisely during the analysis of extra information concerning the construct) and hinder the description process.
We will illustrate this kind of ambiguity by an example encountered during the BMM construct description phase and we will suggest our solution. The example will be based on the "Organization Unit" construct because cetain extra information found about this construct were fairly imprecise.

The ambiguities of this example refer to the information brought out by the partial BMM Meta-Model presented in Figure 7.2. According to the association described in the model, "an Organization Unit *"makes"* an Assessment". As shown in Figure 7.3, there are four categories of assessments: Strength, Weakness, Opportunity, and Threat.
The association *"makes"* doesn't seem appropriate in the present case. Because of the hierarchical relation presented in Figure 7.3, the association implies that an Organization Unit makes a Strength, a Weakness, an Opportunity or a Threat, which is difficult to interpret for the language analyzt. To verify the exact meaning of this assertion, we will analyze the [OMG06] descriptions of Assessment and of one of its four specialized constructs, Strength for instance:

"Assessment: judgement that an influencer affects the employment of means and/or the achievement of ends."

An Assessment is described as a "judgement". It is not wrong to state that "an Organization Unit makes a Assessment" but the term "makes" remains very broad and can lead to various interpretations of the relation.

"Strength: assessment that an influencer indicates an advantage or area of excellence within an enterprise that can impact its employment of means or achievement of ends."

The Strength is described as an Assessment which is, according to its description, a judgement. We infer that a strength is consequently also a judgement. Theoretically, because a Strength is a judgement, it wouldn't be wrong either to use the association *"makes"* between Organization Unit and Strength. However in practice, a great confusion appears in the analyzt's mind when he reads the following relation: "An Organization Unit *makes* a Strength."

This observation leads to conclude that the association *"makes"* is acceptable (although imprecise) between Organization Unit and Assessment, but is awkward while considering Organization Unit and Strength (The case of Strength is applicable to the three other specialized constructs (Weakness, Opportunitie and Threat - for descriptions, see [OMG06])).

The proposal to solve this problem of ambiguity is to replace the term *"makes"* by the term *"appraises"* in the problematical association. Indeed, *"Appraises"* is a much more faithful to the "evaluation" sense that the association aims to express in the relation between Organization Unit and the "Assessment constructs" (Assessment itself, Strength, Weakness, Opportunity and Threat).

This example also demonstrates that some details of the BMM Meta-Model can still be improved.

7.3 BMM constructs mapping

Now, we will put the construct mapping methodology we have defined in chapter 6 into practice on the BMM Organization Unit construct. The description of Organization Unit given by [OMG06] was remastered and a new one was set up during the construct description process (see section 7.1). The remastered description is:

An Organization Unit is a named group of people, eventually part of a bigger group of people, and described by various characteristics. It defines its own purposes, establishes which means to apply to reach its purposes within

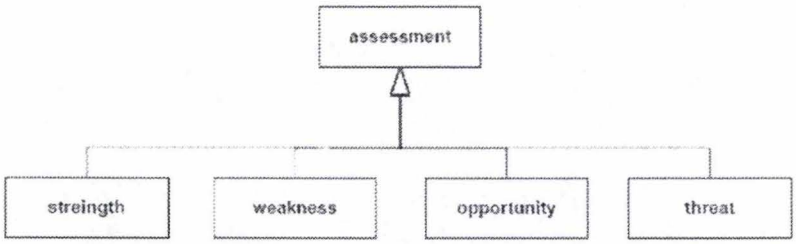

Figure 7.3: The different categories of Assessment (from [OMG06]).

the limits of its budget and decides how to implement those means.

It is this description that will serve as a basis for the construct mapping. We will go successively through the two steps of the methodology to finally obtain the construct mapping:

7.3.1 Step 1

- *What (or Who) does the description aim to describe?*

  The answer is the Organization Unit itself.

  The Organization Unit represents the *"Who"*.

  Are there any details concerning the "Who" in the construct description?

  The answer is yes. They are:

  1. "named";
  2. "group of people";
  3. "eventually part of a bigger group of people";
  4. "described by various characteristics";
  5. "has a budget".

- *What is (are) the primary purpose(s) of the "Who" ?*

  The answer is: "to define its own purpose".

This answer constitutes the *"What"*.

Are there any details concerning the "What" in the construct description?

The answer is no.

- *How does the "Who" reach its primary purpose(s)?*

  The purpose of the *"Who"* is reached by the execution of these two actions:

  1. "to establish which means to apply to reach its purposes";
  2. "to decide how to implement the means".

  Those actions represent the *"How"*.

  Are there any details concerning the "How" in the construct description?

  The answer is yes. The precision concerns the means to apply to reach the purposes that must remain "within the limit of the Organization Unit budget".

- Now, we have built up a set of information units composed of the *"Who"* and its details, the *"What"* and the *"How"* and its details. We have to decide which information units we will keep and which information units we won't take into account in step 2.
  In fact, we will keep all the information units except two of them:

  1. One of the details of the *"Who"*: "An Organisation Unit has a budget".
     We reject this detail because we consider that it does not bring out a significant information to qualify the *"who"*. It can be instead assimilated to the various characteristics of an Organization Unit.

  2. The detail of the *"How"*.
     This detail is not kept because of the first rejection.

  Apart from those two rejections, the other information units form a bunch of selected characteristics of Organization Unit that will be used to define some instances of UEML ontology concepts in step 2.

7.3.2 Step 2

Now, we have to determine for every selected characteristic one or many UEML ontology concepts that will be instanciated to represent the phenomenon described by the characteristic. As explained section 6.2, this operation is subjective.
Instead of describing every instanciation performed for Organization Unit, we will rather focus on one of the seven selected characteristics and see how we decided to represent it in the UEML ontology (the complete construct mapping is depicted in Figures 7.4 and 7.5).

We will choose to analyze the characteristic of the *"What"* emphasized in section 7.3 and represented by "Purpose" in figure 7.4. This characteristic is interesting because it led to a double instancation of ontology concepts (see Figure 7.4 and Figure 7.5).

To understand the instanciations performed, we must consider that the instance of represented class "Organization Unit" mapped onto ActiveThing has already been created (see Figure 7.5).
The section 7.3 defines the *"What"* as follows: "to define its own purpose". The Organization Unit description remastered in section 7.1 states: "An organization unit defines its own purposes". We can infer from those statements that an Organization Unit and its purposes are linked and we want this relation and the purposes to be represented in two distinct ontology concept instances. To find out the right ontology concepts to instanciate, we have to determine how to interpret this relation. The chosen interpretation is that an Organization Unit defines its own purposes in terms of things it wants to act on in order to obtain certain results.

Two ontology concepts seem to match this interpretation. One property (ActingOnRelation) to represent the relation between an Organization Unit and its purposes and one class (ActedOnThing) to represent the things that are acted on.

The UEML ontlogy describes ActingOnRelation as follows:

"A coupling between two things in which the history of the first thing depends on the history of the second thing, but where the second thing's history does not depend on the history of the first."

The UEML ontlogy describes ActedOnThing as follows:

"A changing thing that is acted on by at least one other thing through an acting-on relation, a particular type of coupling. In addition, the acted-on

thing may possess a state law that restricts the combinations of properties that the acted-on thing can possess at the same time."

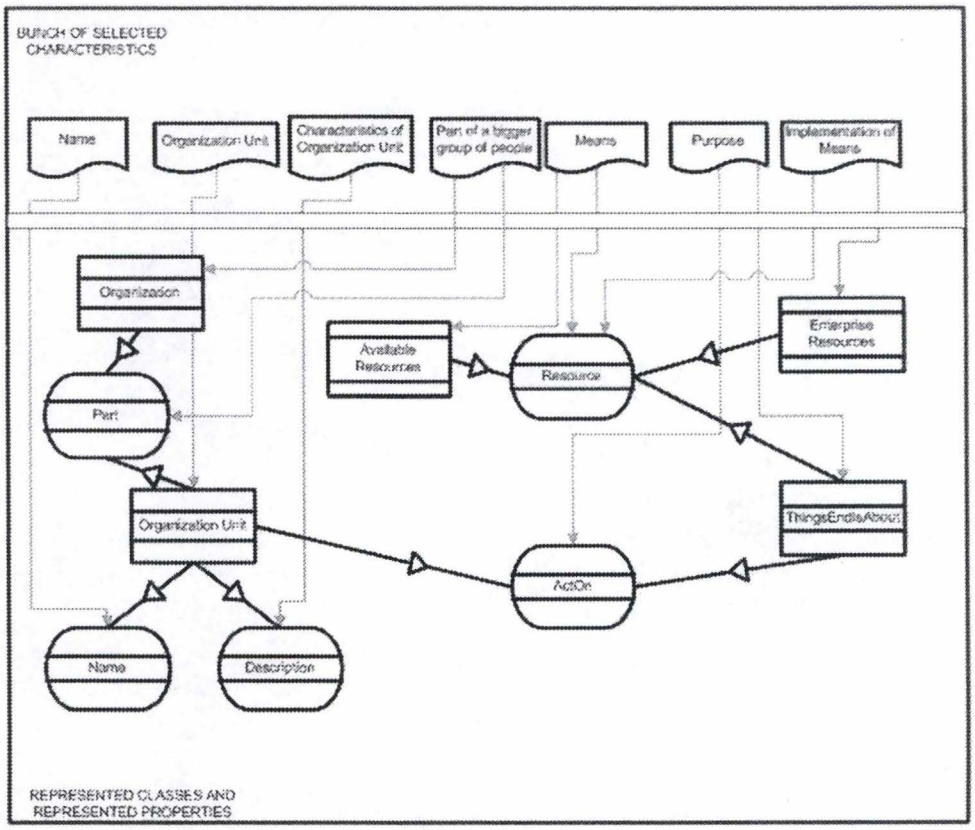

Figure 7.4: The ontological instances corresponding to the selected characteristics of the BMM Organization Unit description.

7.4 Summary

This chapter has shown how to resolve step by step the ambiguities which occured in a specific BMM construct description: Organization Unit. It has also underlined one perfectible aspect of the BMM Meta-Model, that is to say the lack of appropriateness of certain association names, through a specific example. Finally, the last section of this chapter has described the construct mapping performed into the UEML ontology for BMM Organization Unit.

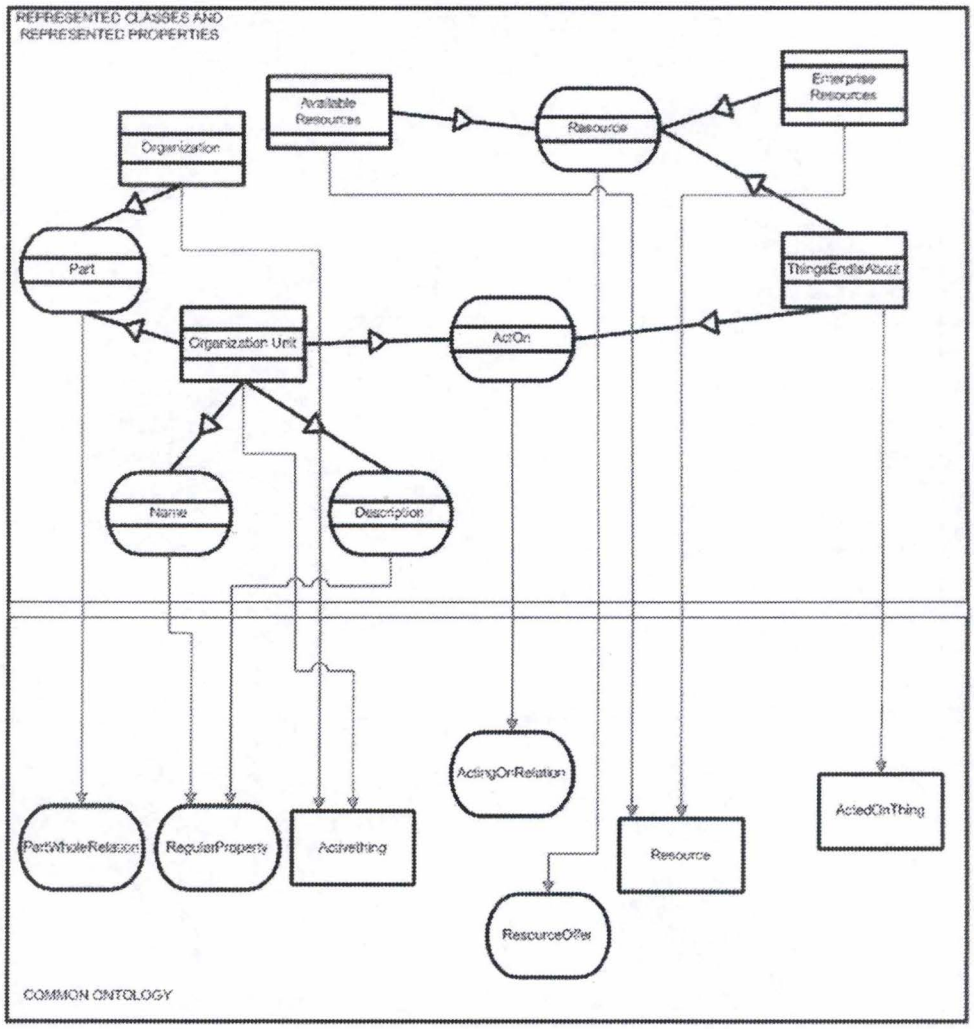

Figure 7.5: Ontological mapping of BMM Organization Unit: the relations between the ontological instances and the ontological concepts that are instanciated.

Chapter 8

*i** analysis

The previous chapter applied the language analysis methodologies described in chapter 6 to perform a BMM analysis in accordance with the UEML approach. This chapter will apply the same makings to present the analysis of a Goal-oriented framework, *i**.

8.1 *i** construct description

The *i** framework was specified for the first time by Eric Yu in his thesis ([Yu95]). Eric Yu is one of the main creators of *i** and [Yu95] is a synthesis that specifies the various features of the *i** framework. Therefore, we will refer to this document as the official *i** specification in the rest of the chapter.

In the light of the numerous *i** constructs, we won't present all the description processes. We will rather proceed as we did in section 7.1 for BMM. We will select a representative construct on the basis of the ambiguities its official description contains. Then we will apply the constructs description methodology presented in section 6.1 to suggest a way to solve these ambiguities and to remaster a new construct description. The other *i** construct mappings are given in Appendix.

The *i** construct we will choose and on which we will perform the construct description process is the *i** "Actor". Indeed, its official description given by [Yu95] contains some ambiguities that deserve to be clarified.

Before starting the construct description process, we must precise that the construct description of "Actor" we will consider is given by [Yu95] in reference to the *i** Strategic Dependency model (see section 5.2.2).

8.1.1 Step 1: Evaluation process

[Yu95] describes an i^* Actor as follows:

"An actor is an active entity that carries out actions to achieve goals by exercising its knowhow."

There are several ambiguities raised up by this description:

1. Who decides which actions the actor must carry out in order to achieve the goals: himself or someone else?
2. What does the description mean by *"goals"*?
3. Who determines the goals to be achieved?

These three ambiguities must be clarified in step 2.

8.1.2 Step 2: Description process

The answer to the first ambiguity is found in a document written, among others, by Eric Yu ([GYV06]) and which states:

"An actor can choose freely among different ways to achieve a goal."

We find out that an Actor decides himself which means to apply in order to achieve the goals. Consequently, the first ambiguity is resolved.

Now, we must answer the following question in order to clarify the second ambiguity: what stands behind "goals"?
The best way to answer this question is to focus on what links two actors in a Strategic Dependency model.

As explained in section 5.2.2, each link between two actors of a Strategy Dependency model indicates that one actor depends on the other for something in order that the former may attain some goal. The depending actor is called the *depender*, the actor who is depended upon is called the *dependee* and the object around which the dependency relationship centres is called the *dependum*.

Therefore, we infer from this explanation that the dependences between actors are means to achieve their "goals". We need to learn more about the nature of these dependencies in order to be able to determine what "goals" means. [Yu95] distinguishes four types of dependencies between actors:

"Dependency Types: We distinguish among four types of dependencies based on the type of the dependum. In world modelling, it has been found useful to distinguish among three basic ontological categories: entities, activities, and assertions. Entities are used to model objects in the world. These can be physical or informational. Activities produce changes in the world. An assertion expresses a state or condition about the world. From these basic categories we get three types of intentional dependencies: Resource dependency, Task dependency and Goal dependency. A fourth type which we call Softgoal dependency is based on a notion of non-functional requirements (or quality requirements) in software engineering."

Thanks to the four types of dependencies highlighted by [Yu95], we are now able to determine the purposes of the dependencies:

1. in a Resource dependency, the purpose of the dependency is a Resource; a Resource in *i** being "A physical or informational entity needed to achieve some goal or to perform some task [GYV06]."

2. in a Task dependency, the purpose of the dependency is a Task; a Task in *i** being "A course of action to be carried out [GYV06]."

3. in a Goal dependency, the purpose of the dependency is a Goal; a Goal in *i** being "A condition or state of affairs to be achieved [GYV06]."

4. in a Softgoal dependency, the purpose of the dependency is a SoftGoal; a SoftGoal in *i** being "A goal without a clear-cut criterion for achievement [GYV06]."

Therefore, the "goals" encompass either a Resource, a Task, a Goal and a SoftGoal. Consequently, the statement "to achieve goals" of the initial description can be replaced in any case by one (or several) of the following statements:

1. "to furnish a Resource";

2. "to perform a Task";

3. "to attain a Goal";

4. "to attain a SoftGoal".

The second ambiguity is now clarified.

Futhermore, we also came across the answer to the third ambiguity while searching the answer to the second ambiguity. Indeed, we discovered that a *depender* (which is an Actor) needs a *dependee* (which is also an Actor) to

achieve a *dependum*. Although the *depender* does not choose how to reach some "goals", it asks and counts on the *dependee* to furnish a Resource, to perform a Task, to attain a Goal or to attain a SoftGoal. Now, we can state that the Actor himself (the *depender* to be specific) determines the "goals" to achieve. The third ambiguity is solved.

Having clarified all the ambiguities raised in step 1, we are now able to remaster a new construct description of "Actor" which will take the answers brought out in step 3 into account. Here is the remastered and final description:

"An actor is an active entity which chooses freely among different actions and carries them on by using its knowhow in order to attain a goal (or a softgoal), to perform a task or to deliver a resource."

8.2 *i** construct mapping

In the previous section, we resolved three ambiguities found in the *i** Actor description and we remastered a new description thanks to the construct description methodology. Now, we will see how the new description is split up into several information units and then how the most relevant information units are mapped into the UEML ontology. Those two steps are based on the construct mapping methodology described in section 6.2.

8.2.1 Step 1

This step will decompose the new Actor description by using several questions in order to form a bunch of basic characteristics concerning the construct.

- *What (or Who) does the description aim to describe?*

  The answer is the "actor".
  The "actor" represents the *"Who"*.

  Are there any details concerning the "Who" in the construct description?

  The answer is yes. They are:

  1. "...is an active entity...";
  2. "...by using its knowhow...".

- *What is (are) the primary purpose(s) of the "Who" ?*

  The answers are:

  1. "to attain a goal (or a softgoal)";
  2. "to perform a task";
  3. "to deliver a resource".

  Those answers constitute the *"What"*.

  Are there any details concerning the "What" in the construct description?

  The answer is no.

- *How does the "Who" reach its primary purpose(s)?*

  The answer is: "(An actor) chooses (freely) among different actions and carries them on...".
  This answer represents the *"How"*.

  Are there any details concerning the "How" in the construct description?

  The answer is yes.
  The detail concerns the way an actor chooses among the actions that it performs, that is to say "freely".

- Now that we have emphasized a set of information units concerning *i** Actor, we must decide which information units will be kept to perform the Actor mapping into the UEML ontology in step 2.
  Actually, every information unit brings out some relevant precisions concerning the construct. Therefore, all of them will be kept and will be gathered together to form a bunch of basic characteristics of Actor that will be used to perform the ontological mapping of the construct.

8.2.2 Step 2

In the previous step, we gathered together some basic characteristics of the *i** Actor description. The purpose of the current step will be to create, for each basic characteristic, one or several instances of UEML ontology concepts. That will describe the phenomena represented by Actor.

We will proceed as we did for BMM Organization Unit in section 7.3.2. Instead of considering all characteristics emphasized in step 1, we will present the mapping of Actor through the case of a selected charateristic. We will figure out how this characteristic is described into the UEML ontology.

First, we must select one of the basic characteristics of the Actor description. We will choose "carries on actions". We consider it to be a good candidate to be analyzed because, as we will see later, it generates no less than four instances of UEML ontology concepts. Therefore, "carries on actions" is a significant characteristic of the Actor description.

We saw in section 8.2.1 that "carries on actions" represents the *"How"*. It is a mean to perform a Task, to achieve a Goal (or a SoftGoal) or to furnish a Resource. The interpretation we give to "carries on actions" is the following: performing some actions in order to reach a purpose implies that some "things" concerning the purpose (in the present case, some "things" concerning a Task, a Goal (or a SoftGoal) or a Resource) must be acted on.

Three UEML ontology concepts are necessary to describe this interpretation (see Figure 8.2). They are represented by four ontology instances (see Figure 8.1). The three UEML ontology concepts are:

1. **"ActingOnRelation"**: this ontology property is needed in order to represent a common property that is shared by all the "things" on which an Actor acts on, typically the Resources and the "things" which concern a Goal. It is represented by the instance *ActOn*;

2. **"ActedOnThing"**: this ontology class is needed in order to represent the "things" which are concerned by a Goal (or a Task) and on which an Actor acts on. it is represented by the instances *ThingGoalIsAbout* and *Resource*;

3. **"MutualProperty"**: this ontology property is needed in order to represent a common property which links a Task and the Resources needed to perform the Task. It is represented by the instance *IsAbout*.

8.3 Summary

This chapter discussed the i* analysis through a example of construct: *i** Actor. First, it underlined several ambiguities in the construct description and brought out some solutions. Then, it proposed an UEML ontology mapping realized on the basis of a remastered construct description.

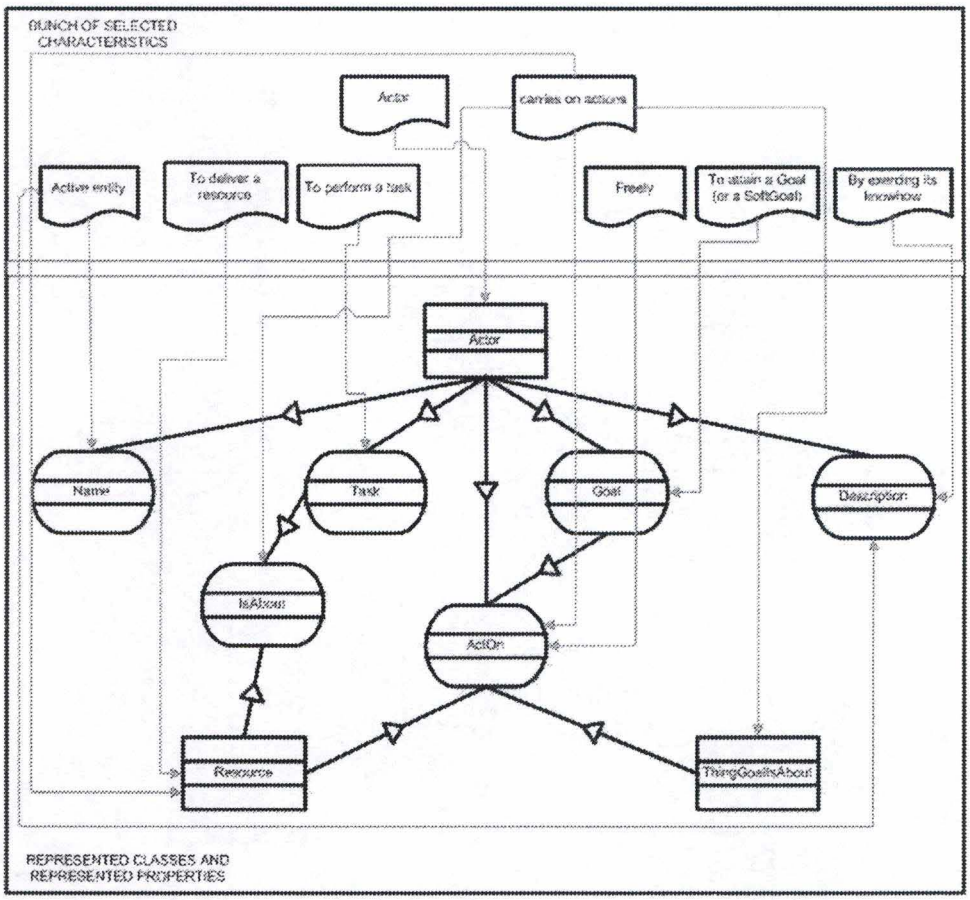

Figure 8.1: The ontological instances corresponding to the selected characteristics of the i^* Actor description.

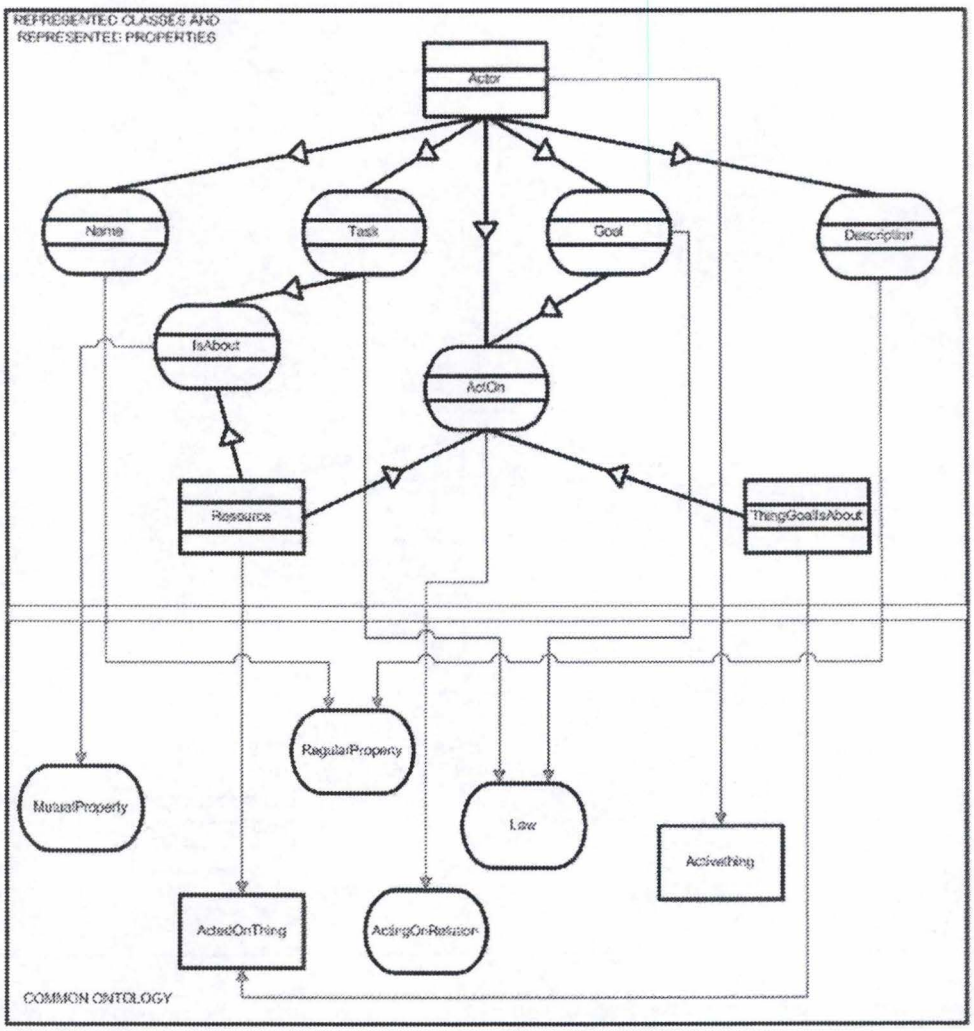

Figure 8.2: Ontological mapping of i^* Actor: the relations between the ontological instances and the ontological concepts that are instanciated.

Part III

Evaluation

Chapter 9

Constraints validation

The objective of this chapter is to present a constraint validation that was performed on the BMM and i^* analysis presented in chapter 7 and 8. This validation aims to allow the improvement of language analysis by checking their consistency thanks to a set of predefined constraints. The tool used to perform these validations is called "UEML Validator" (see section 4.2). As an example, we will present the results generated by the tool on a chosen BMM construct and we will interpret those results. Then, we will propose a solution to resolve the inconsitencies underlined by the results. Finally, we will provide a short evaluation of the UEML Validator.

9.1 Validation of BMM

This section will set out a specific BMM constraints validation. Indeed, it will neither present nor analyze all the BMM constraints validations. BMM construct analysis are numerous and it would treat several times the same cases of inconsistency; that would not be relevant. The section will rather focus on the BMM Organization Unit analysis expounded in sections 7.1 and 7.3, find out which inconsistencies arise and resolve them.

9.1.1 Presentation of the results obtained

The "UEML Validator" applied to the Organization Unit analysis has generated the two kinds of inconsistency:

1. the first kind of inconsistency concerns each represented property as well as each represented class of the Organization Unit represented phenomena. One of the results produced by the "UEML validator" in regard to this sort of inconsistency is:

"Represented phenomenon BMM_OUClassRole_AvailableResource plays no role."

2. the second kind of inconsistency concerns once again each represented property as well each represented class of the Organization Unit represented phenomena. The "UEML validator" describes those inconsistencies by producing this kind of result:

   "Represented class BMM_OUClassRole_OrganizationUnit possesses represented property BMM_OUPropertyRole_Name, but no corresponding ontology class possesses a corresponding ontology property."

9.1.2 Interpretation of the obtained results

Now, we will interpret the two results given in the previous section.

1. The first result claims that a Represented phenomenon does not play a "role".
   We want to learn about the concept of "role" in order to understand the object of the inconsistency.

   [Mah06] describes all the attributes that a represented phenomenon must have. It states that a represented phenomenon must have, among other attributes, a *"roleName used to name the phenomenon"*.
   Futhermore, two constraints taken into account by the "UEML Validator" are especially interesting because they deal with the notion of "role":

   - *"If a ConstructDescription contains more than one RepresentedClass, each of them must have a "roleName" that is unique to the ConstructDefinition. [Mah06]"*
   - *"If a RepresentedClass has more than one RepresentedProperty, each of them must have a "roleName" that is unique relative to the RepresentedClass. [Mah06]"*

   Having discovered that every represented phenomenon must have a name (called "RoleName") which must respect both constraints, we will now verify if every represented phenomenon of Organization Unit meets this condition.

   The answer is that none of them has a name. Therefore, we discover the reason of the first inconsistency.

2. The second result leads us to consider two specific constraints which deals with the relation between a represented phenomenon and the ontology concept it instanciates:

   - *"If a RepresentedClass has a RepresentedProperty, the corresponding ontClass must have the corresponding ontProperty as "characteristic".[Mah06]"*
   - *"Conversely, if a RepresentedProperty has a RepresentedClass, the corresponding ontProperty must be "characteristic" of the corresponding ontClass.[Mah06]"*

   Thanks to those constraints, we are now able to interpret the second inconsistency. The problem is the following:

   the represented property `"BMM_OUPropertyRole_Name"` characterizes the represented class `"BMM_OUClassRole_OrganizationUnit"`.
   As `"BMM_OUClassRole_OrganizationUnit"` is mapped onto *ActiveThing* and `"BMM_OUPropertyRole_Name"` onto *RegularProperty*, *RegularProperty* has to characterize *ActiveThing*, which is not the case.

9.1.3 Solutions to resolve the inconsistencies

1. The solution to solve the first inconsistency consists in assigning a name to each represented phenomenon by taking into account the name of the constraints defined in the previous section. As shown in Figure 9.1, the name assignment can be done with the Protégé tool, via the tab of the represented phenomenon.

2. The solution to solve the second inconsistency consists in verifying if an ontology class possesses a specific ontology property or if an ontology property characterizes a specific ontology class. If the ontology class does not possess the specific ontology property (or if the ontology property does not characterize a specific ontology class), the relation of possession (or of characterization) must be created between the two ontology concepts. Those operations of verification and creation can be managed with the Protégé tool. Figure 9.2 shows the creation of a relation of possession between the ontological class *ActiveThing* and the ontological property *RegularProperty*.
   However, modifying the ontology must always be considered as a borderline solution. We propose this solution because, in the present case, it is actually difficult to translate the constraint in terms of solutions. We will discuss this problem in section 9.2.

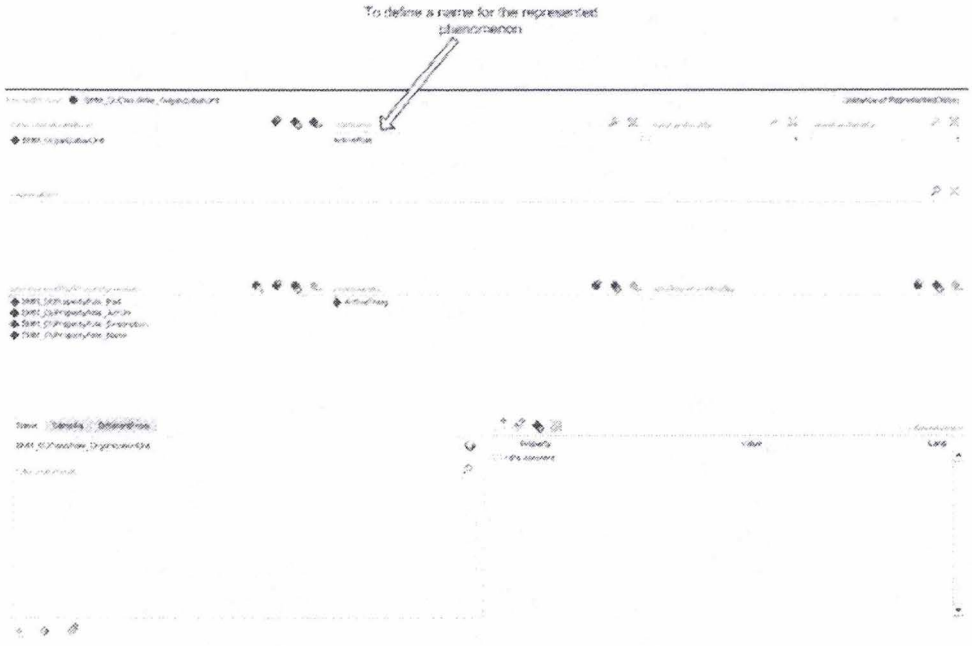

Figure 9.1: A name assignment in the tab of a represented class Organization Unit.

9.2 evaluation of the UEML Validator

This section aims to underline a possible improvement that could be brought to the UEML Validator. As pointed out in the previous section, a inconsistency generated by the tool can sometimes be difficult to interpret and therefore to solve. A possible improvement could be to redefine certain results generated by the tool in order to make them more understandable and also to propose some possibilities to solve the inconsistencies emphasized by the tool in order to assist the user.

9.3 Summary

This chapter presented two examples of inconsistencies found by the UEML Validator on the analysis of BMM. We proposed an interpretation and a possible solution to solve them. We conclude the chapter by pointing out a possible improvement for the tool.

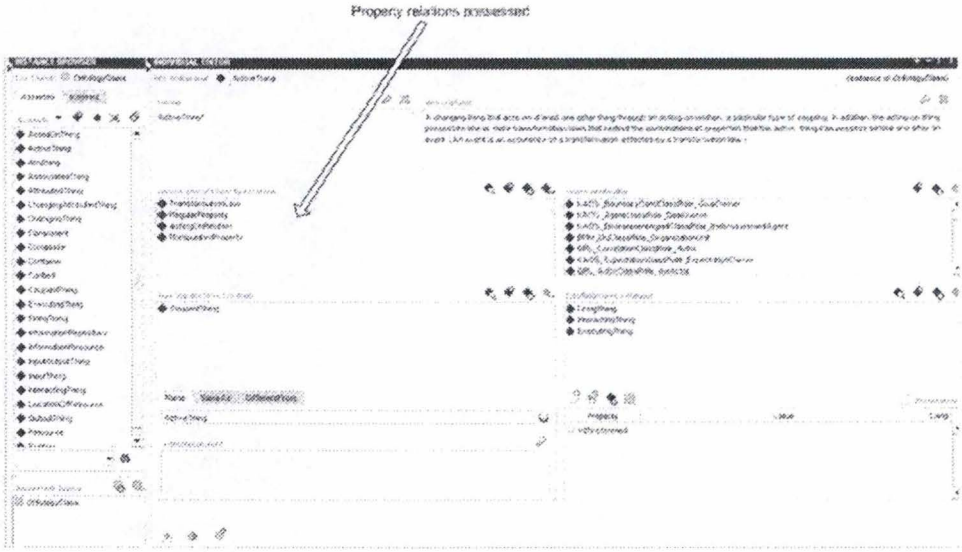

Figure 9.2: The establishement of a relation of possession between the class *ActiveThing* and the property *RegularProperty*.

Chapter 10

Languages comparison

This chapter aims to test out a construct comparison tool, the similarity Plug-In, presented in section 4.3. The chapter will evaluate the relevance of the results provided by this tool on the basis of two comparisons: one between BMM and i^* and another one between i^* and GRL. The point in performing these two comparisons will be to evaluate the relevance of the results produced by the tool from different EML approaches. On one hand a comparison between a Business Oriented EML (BMM to be specific) and a Goal oriented EML (i^*) and on the other hand a comparison between two Goal Oriented EMLs (i^* and GRL to be specific). The first section of the chapter will define a comparison methodology that we will use for both comparisons. The second section will set out the comparison between BMM and i^*. The third section will expound the comparison between i^* and GRL. Finally, the fourth section will provide a evaluation of the similarity Plug-In on the basis of the results obtained in the second and third sections.

10.1 Comparison methodology

The similarity Plug-In actually allows four kinds of comparisons:

1. a comparison between two constructs that belong to the same language;
2. a comparison between two constructs that belong to different languages;
3. a comparison between one construct of a language and all the constructs of the same language;
4. a comparison between one construct of a language and all the constructs of another language.

In the present case, we will perform the language comparisons in accordance with the fourth kind of comparison enabled by the tool. The reason

for this choice is that, for each comparison performed, the fourth kind of comparison will provide us a broad set of results and therefore will allow us to verify the relevance of more than one result. The third kind of comparison would also allow us to proceed that way but, as said previously, we want to apply the tool on different EML approaches.

We now need to set up a comparison methodology that implements precisely this approach of language comparison and that we will use in sections 10.2 and 10.3 to perform the comparisons. The approach we propose goes successively through two steps:

- **Step 1: selection of constructs.**

  The first step consists of selecting one or several constructs of the first language (in the rest of the document, they will be named **"selected"** constructs). They are chosen because they are most likely to be similar to one or many constructs of the second language (the latter will be named **"possibly similar"** constructs). The chances for a construct of a given language to be similar to a construct of another language are determined by two criteria:

  1. *the name of the constructs.* If two constructs have the same name, there is a chance that they present some similarities.
  2. *the decription of the constructs.* If a construct description bears some resemblance to another construct description, there is a chance that they present some similarities.

  If a construct of the first language (called X for instance) respects at least one of these two critera with regard to one or several constructs of the second language (let Y,Z be those constructs for instance), we will consider that X is a **"selected"** construct potentially related to two **"possibly similar"** constructs: Y and Z.

  When this selection is made (see Table 10.1), we have to perform a comparison between each selected construct of the first language and all the constructs of the second language (see Figure 10.1). This comparison is performed thanks to the similarity Plug-In.

- **Step 2: analysis of the comparison results.**

  When comparing a selected construct with all the constructs (let all the constructs be N constructs for instance) of the second language, the similarity Plug-In provides in fact N results; one for each comparison between the **"selected"** construct and each of the constructs of

"Selected" constructs Language Alpha	**"Possibly similar" constructs Language Beta**
construct V	*construct W*
construct X	*construct Y, construct Z*

Table 10.1: Example of the **"selected"** constructs of a hypothetical language Alpha, each of them corresponding to one or many **"possibly similar"** constructs of another hypothetical language Beta.

the second language. A result is always contained between 0 and 1. A result of 0 means that the **"selected"** construct has nothing in common with the construct of the second language. Conversely, a result of 1 means that the **"selected"** construct is perfectly similar to the construct of the second language.

When using the similarity Plug-In to compare two constructs, we need to set a threshold in order to interpret the result obtained. We actually need a threshold to determine, on the basis of a result, if a construct is highly similar to another. This threshold will be contained between 0,5 (the half similarity) and 1 (the perfect similarity). The more a threshold is close to 1, the more representative of a high similarity it is but at the same time, it gets more restrictive. We will decide to set the threshold to 0,8. Although 0,8 is close to 1, it is not too restrictive. Consequently, we will consider that two constructs are highly similar if the comparison between them reveals a result $\geq 0,8$.

Therefore, when analysing a specific result obtained, two cases concerning a **"possibly similar"** construct can occur:

1. the comparison reveals that the **"possibly similar"** construct emphasized in step 1 is highly similar to its related **"selected"** construct (result $\geq 0,8$). This noticing leads to temporaly confirm the initial similarity hypothesis.
2. the comparison shows that there are some differences between the **"possibly similar"** construct emphasized in step 1 and its related **"selected"** construct (result $\prec 0,8$). This situation leads to temporaly invalidate the initial similarity hypothesis.

Finally, we will have to determine the relevance of the result provided by the tool by confirming (or invalidating) permanently the initial similarity hypothesis between the two constructs. This can be done by answering the following question: *On the basis of the ontological representation of the constructs we are comparing, do we agree with the*

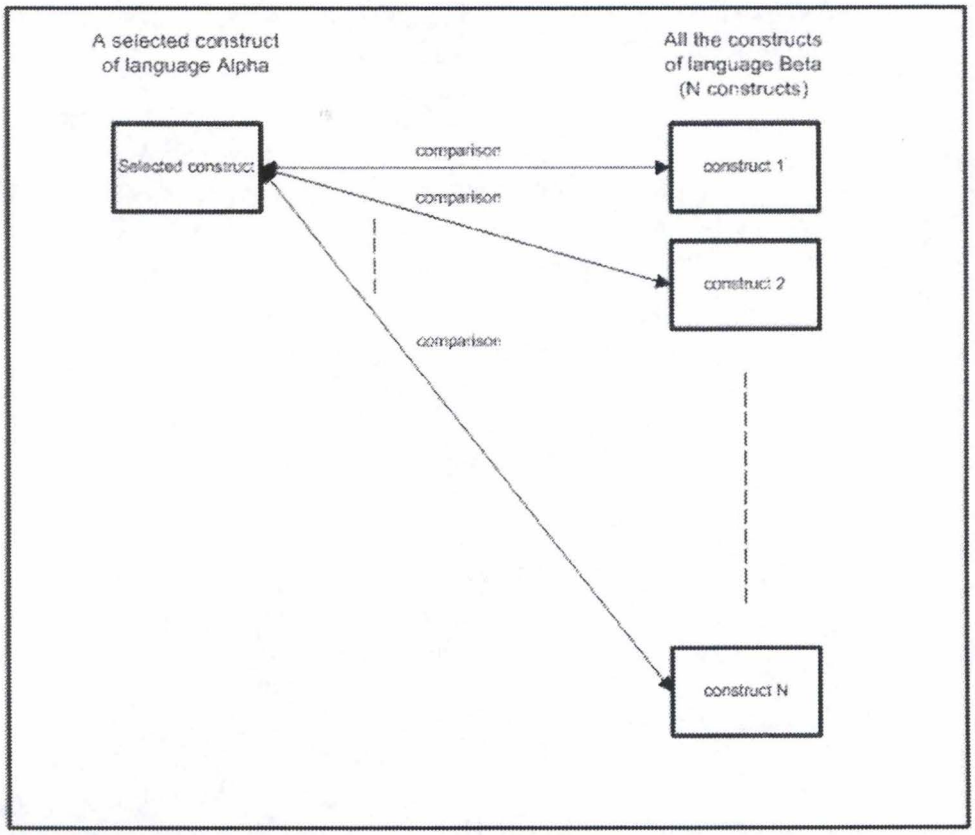

Figure 10.1: After having been **"selected"** and potentially related to one or many **"possibly similar"** constructs of a second language, a **"selected"** construct is compared with the N constructs of the second language.

similarity result obtained? This question will be answered by analysing the ontological representation of each construct we compared and by determining if they are in accordance with the result obtained. There is a part of subjectivity in the answer, but it must always be based on an argumentation.

10.2 Comparison BMM - *i**

In this section, we will perform some comparisons between a set of **"selected"** BMM constructs and the *i** constructs. Then we will find out if the hypothetical similarities between the **"selected"** BMM constructs and their **"possibly similar"** *i** constructs are justified. To carry out those comparisons and evaluations, we will put the methodology described in section 10.1 into practice.

10.2.1 Step 1: selection of constructs

According to the two critera of potential similarity defined in the previous section, three BMM constructs are likely to be similar to certain *i** constructs.

- **BMM Organization Unit** (selected on the basis of the description criterion)

  The description of Organization Unit shares some concepts with the description of *i** Actor. The description of Organization Unit and of Actor are respectively given in section 7.1.2 and in section 8.1.2. They have mainly three characteristics in common:

  1. both are active entities;
  2. both want to reach some purposes;
  3. both choose freely among different means to apply in order to reach their purposes.

  Certain *i** constructs present some close relationships with *i** Actor and therefore deserve to be considered within the context of a comparison with BMM Organization Unit.

  Those constructs are *i** Agent, *i** Position and *i** Role. [Yu95] describes Agent, Position and Role as *"the various more specialized notions of actors"* and adds *"When we separate out these components of a social actor, each component is a partial description of the social actor. Each component gives some hints about how the social actor might behave, or is expected to behave, in some specialized, narrow context"*. As specialized notions of Actor, these three constructs will also be taken into account during the analysis of the comparison results in step 2.

  Therefore, Organization Unit will be a **"selected"** construct which is potentially related to four **"possibly similar"** constructs: Actor, Agent, Position and Role.

- **BMM Goal** (selected on the basis of the name criterion)

  BMM Goal has the same name as an *i** construct: *i** Goal.

  Therefore, BMM Goal will be a **"selected"** construct which is potentially related to a **"possibly similar"** construct: *i** Goal.

- **BMM Business Process** (selected on the basis of the description criterion)

  [OMG06] describes the BMM Business Process as follows: *"A unit of work to accomplish a transformation of information or resources contributing to the business objective of a conventional or orchestrated process."*

  Certain elements of the *i** Task description given by [GYV06] seem to match the BMM Business Process description.

  [GYV06] describes the *i** Task as follows: *"A course of action to be carried out. It specifies a particular way of doing something, typically to achieve some goal."*

  Through those descriptions, the commonalities between the two constructs can be summarized by the following points:

  1. both constructs represent an action that has to be accomplished.
  2. both contribute, through their realization, to the achievement of a certain purpose.

  Therefore, BMM Business process will become a **"selected"** construct and will be potentially related to *i** Task.

The three **"selected"** BMM constructs and their potentially related *i** constructs are depicted in Table 10.2

Selected constructs BMM	**"Possibly similar" constructs *i****
Organization Unit	*Actor, Agent, Position, Role*
Goal	*Goal*
Business Process	*Task*

Table 10.2: The **"selected"** BMM constructs and their corresponding **"possibly similar"** *i** constructs.

10.2.2 Step 2: analysis of the comparison results

Comparison: BMM Organization Unit - *i** constructs

The results of the comparison between BMM Organization Unit and all the *i** constructs are shown in Table 10.3.

i^* constructs	Results of the comparison with BMM O.U.
Actor	*0,69*
Task	*0,69*
SoftGoal	*0,62*
Goal	*0,59*
Resource	*0,58*
Actor association link	*0,58*
Dependency link	*0,58*
Means ends link	*0,55*
Position	*0,55*
Decomposition link	*0,49*
Agent	*0,19*
Role	*0,18*
Contribution link	*0,18*
Belief	*0,13*

Table 10.3: Results of the comparison between BMM Organization Unit and all the i^* constructs

The results expounded in Table 10.3 reveal two important information:

1. the i^* Actor is actually the most similar i^* construct in comparison with BMM Organization Unit. However, its result is $\prec 0,8$, which means that, in regard to the similarity threshold set in the methodology (see section 10.1), some obvious differences between the ontological representations of the two constructs exist according to the score obtained. On the basis of this result, the initial similarity hypothesis between the two constructs is temporaly invalidated.

2. Position, Agent and Role have a very low score with respectively 0,55, 0,19 and 0,18. Consequently, none of them can be considered as highly similar constructs in comparison with Organization Unit. The initial similarity hypothesis is also temporaly invalidated.

Now, we will analyse the ontological representations of Actor, Agent, Position and Role and we will determine if the differences that exist with the Organization Unit ontological representation are sufficient to invalidate permanently the similarity hypothesis.

The mappings of BMM Organization Unit and of i^* Actor into the UEML ontology are respectively represented in Figure 7.5 and in Figure 8.2. The mappings of i^* Agent, i^* Position and i^* Role are represented in Appendix.

When analysing the mappings, we find out some noticable differences in the way the constructs are represented in the common ontology.

- The i^* Actor is represented into the common ontology by decomposing the purposes that it aims to reach into "Task" and "Goal" (two represented properties, both representing *law*) and "Resource" (a represented class representing *ActedOnThing*).
  Conversely, the Organization Unit ontological representation suggests another modelling approach. It rather focus on the representation of the elements concerned by the purposes to reach with "ThingEndIsAbout" (a represented class representing *ActedOnThing*) and on the resources needed to implement the means used to reach the purposes with "Available Resources" and "Enterprise Resources" (two represented classes representing *resource*).
  In this case, the score is justified. Therefore, we will invalidate permanently the similarity hypothesis.

- As specified in section 10.2.1, Agent, Position and Role are specialized notions of Actor. The reason why their similarity score is low is precisely because their ontological representation focus on the specificities that each of those specialized constructs brings in regard to i^* Actor, rather than focusing on the basic features of i^* Actor. For instance, the ontological representation of Role only focuses on the behavioural aspect of Actor without re-modelling the purpose pursued by Actor.
  On a sheer aspect of ontological representation, we will accept those scores and invalidate permanently the similarity hypothesis. But in section 10.4.1, we will evoque a problem raised up by this case; the problem to represent the specializations of constructs in the Protégé tool.

- A last insteresting observation about the comparison results concerns the i^* Task, which ontological representation is given in Appendix. We were actually not expecting i^* Task to be as similar as i^* Actor in the comparison with Organization Unit. Indeed, its similarity score (0,69) is exacltly the same as the one of i^* Actor. However, they do neither have the same construct name nor similar descriptions.
  The reason of this identity is that the ontological representation of i^* Task needs the same ontological concepts as those of i^* Actor. The difference between the two representations actually lies in the fact that the represented properties and the represented classes that instanciate the same ontological concepts are not related in the same way in both representations. For instance, "Description" and "Name" (two represented properties representing *RegularProperty*) are related to "Task" (a represented property representing *law*) and not to "Actor" as it is the case in the Actor representation.
  This issue constitutes a basis for a possible improvement of the similarity Plug-In. It will be discussed later on in section 10.4.1.

Comparison: BMM Goal - i^* constructs

The results of the comparison between BMM Goal and all the i^* constructs are shown in Table 10.4.

i^* constructs	Results of the comparison with BMM Goal
Actor	*0,94*
Task	*0,94*
Goal	*0,84*
Means ends link	*0,77*
Position	*0,77*
SoftGoal	*0,77*
Actor association link	*0,71*
Resource	*0,71*
Dependency link	*0,71*
Decomposition link	*0,67*
Role	*0,29*
Contribution link	*0,29*
Belief	*0,29*
Agent	*0,23*

Table 10.4: Results of the comparison between BMM Goal and all the i^* constructs

The results presented in Table 10.4 reveal two important information:

1. i^* Goal reaches a similarity score $\geq 0,8$ (0,84 to be specific). Therefore, according to the similarity threshold, we can consider it as a highly similar construct in comparison with BMM Goal. This result temporaly confirms the hypothesis of similarity made in step 1.

2. Although BMM Goal and i^* Goal share some similarities, the results demonstrate that i^* Actor and i^* Task are even more similar to BMM Goal (both have a result of 0,94). This result was unexpected and deserves to be investigated.

Now, we will analyse the ontological representations of i* Goal and we will determine if similarities with the Organization Unit ontological representation are sufficient to validate permanently the similarity hypothesis. We will also analyse the ontological representations of i^* Actor and i^* Task in

order to investigate the unexpected result.

The construct mappings of BMM Goal, i^* Goal and i^* Task are depicted in Appendix. The mapping of i^* Actor is represented in Figure 8.2.

After having analysed the different mappings, we can emphasize the following observations:

- concerning the first information, the analysis of the ontological representation of BMM Goal and of i^* Goal confirms both representations are very similar. The major difference between them lies in the represented property "Goal" which instanciates *StateLaw* in BMM and *Law* in i^*. Therefore, we can confirm the hypothesis of similarity made in step 1.

- concerning the second information, we conclude that the differences between the ontological representations of those constructs and of BMM Goal are actually pretty slight. The tiny differences between the similarity scores of i^* Goal (0,84) and of i^* Actor (or i^* Task) (0,94) are explained by an observation concerning the ontological concepts instanciated: BMM Goal, i^* Actor and i^* Task instanciate *MutualProperty* to represent a shared property. In a similar context, i^* Goal instanciates *ActingOnRelation* instead. *ActingOnRelation* specializes *MutualProperty* in the UEML ontology. This explains the slightly different score in favour of i^* Actor and i^* Task.

Comparison: BMM Business Process and i^* constructs

The results of the comparison between BMM Business Process and all the i^* constructs are shown in Table 10.5.

The results depicted in Table 10.5 reveal two important information:

1. i^* Task defined in step 1 as a **"possibly similar"** construct potentially related to BMM Business Process finally reaches a score $\prec 0,8$ (0,73 to be specific). This score does not attain the minimum threshold. Therefore, we will temporaly invalidate the hypothesis of similarity made in step 1 and we will analyse the ontological representation of both constructs later on.

2. The results of the comparison reveal that the most similar constructs are actually two types of links: Decomposition link and Means ends link. We will also find out why they are so similar to BMM Business Process.

*i** constructs	Results of the comparison with BMM B.P.
Decomposition link	*0,87*
Means ends link	*0,87*
Goal	*0,82*
Actor association link	*0,81*
Dependency link	*0,81*
Position	*0,76*
SoftGoal	*0,76*
Actor	*0,73*
Task	*0,73*
Resource	*0,51*
Belief	*0,33*
Contribution link	*0,33*
Role	*0,25*
Agent	*0,19*

Table 10.5: Results of the comparison between BMM Business Process and all the *i** constructs

The construct mappings on which the analysis will be based (the mappings of *i**Task, BMM business process, *i** Decomposition link and *i** Means ends link to be specific) are set out in Appendix.

- With regard to the differences between BMM Business Process and *i** Task, the analysis of their respective ontological representations reveals two interesting obsevations.

  The first observation concerns a noticable difference between the ontological representations. The representation of Business Process models a core represented property (named "BusinessProcess") which represents *TransformationLaw*. In other words, that aims to express that Business Process implies a transformation of "things" in its process. Conversely, the representation of Task does not express such a characteristic. It simply expresses that some operations have to be carried out without specifying any kind of transformation to perform. Therefore, the representation of Task models a represented property (named "Task") which represents *law*.

  The second observation concerns the represented property "CoA" of BMM Business Process and the represented property "Description" of *i** Task. They both aim to model a specification of the two properties emphasized in the first observation but actually they are not mapped onto the same ontological concept. "Description" is mapped onto *Reg-*

ularProperty while "CoA" is mapped onto *Statelaw*. This difference of mapping is due to the fact that "Description" implies various characteristics, possibly of different nature, concerning Task while "CoA" only implies a precise specification of how the Business Process has to be carry out.

Those two explanations justify the similarity score obtained. Therefore, we will invalidate permanently the similarity hypothesis.

- Concerning *i** Decomposition link and *i** Means ends link, they have a good similarity score simply because they instanciate a few identical (sometimes more general) UEML concepts in comparison with Business Process. But basically they don't aim to represent an identical concept, consequently they are not related in the same way. However that is not taken into account by the similarity Plug-In. This issue will be discussed in section 10.4.1.

10.3 Comparison *i** - GRL

This section will expound some comparisons between two goal oriented EMLs: *i** and GRL. The comparisons will be carried out on the basis of the comparison methodology set out in section 10.1.

10.3.1 Step 1: selection of constructs

We will first select some candidates among the *i** constructs and we will relate each of them to one or many **"possibly similar"** GRL constructs.

In the present case, we will only use the selection criterion based on the name of the constructs (see section 10.1). Indeed, we don't need to resort to the criteron based on similarities between construct descriptions because actually, a lot of *i** and GRL constructs have the same name. Consequently, the criteron based on the name of the constructs will be enough to find some candidates.
However, the criteron is not sufficient to perform the construct selection itself. Because of the name similarities between *i** and GRL constructs, the potential candidates are numerous and it would be too long to consider them all in this section. Therefore, we will have to perform the selection of *i** constructs based on our own perception.

We will choose *i** Actor, *i** Goal and *i** Task to become the **"selected"** constructs that will be compared with all the GRL constructs. All of them were **"possibly similar"** constructs related to three different BMM constructs in the last section. Choosing them for the comparison with GRL

constructs is relevant because it will create a logical continuity in regard to the comparisons carried out in the previous section.

As shown in Table 10.6, each of these *i** constructs will be respectively related to **"possibly similar"** GRL Actor, GRL Goal and GRL Task.

Selected constructs *i**	**"Possibly similar" constructs GRL**
Actor	*Actor*
Goal	*Goal*
Task	*Task*

Table 10.6: The **"selected"** *i** constructs and their corresponding **"possibly similar"** GRL constructs.

10.3.2 Step 2: analysis of the comparison results

Having selected three *i** constructs and having related them to some **"possibly similar"** GRL constructs, we will now successively analyse the results of the comparisons between each *i** **"selected"** constructs and the GRL constructs. Then, we will infer some observations on the basis of the scores obtained.

Comparison: *i** Actor - GRL constructs

The results of the comparison between *i** Actor and all the GRL constructs are depicted in Table 10.7.

On the basis of the results presented in Table 10.7, two observations deserve to be underlined:

1. GRL Actor only reaches a similarity score of 0,64 in comparison with *i** Actor. This score is noticeably under the similarity threshold set at 0,8. In theory, that makes GRL Actor fairly different from *i** Actor. In practice, we will investigate the differences that exist between their ontological representation and explain them. But first, we will temporaly invalidate the hypothesis of similarity made in step 1.

2. Unexpectedly, GRL Goal and GRL SoftGoal are perfectly similar to *i** Actor. We will also find out the reason of this similarity.

GRL constructs	Results of the comparison with i^* Actor
SoftGoal	*1*
Goal	*1*
Resource	*0,75*
Actor	*0,64*
Contribution	*0,61*
Means ends	*0,56*
Decomposition	*0,56*
Dependency	*0,51*
Correlation	*0,51*
Task	*0,49*
Belief	*0,23*

Table 10.7: Results of the comparison between i^* Actor and all the GRL constructs

The analysis of the results will be based on the ontological mappings of i^* Actor, GRL Actor, GRL Goal and GRL SoftGoal. All of them are represented in Appendix.

- Concerning the first observation, the ontological mapping of GRL Actor reveals that in fact, this construct has a pretty similar way to represent the concept of "actor" in comparison with i^* Actor. They both represent the resources needed to perform the tasks, the tasks to be carried out by the actor, the things needed to achieve a goal and the goals to be reached by the actor. Actually, the low score is merely due to the slight differences in the ontological concepts instanciated. For instance, in the GRL Actor representation, a task represents a *stateLaw* while in the i^* Actor it represents a *law*. Moreover, the ontological representation of GRL Actor instanciates a *RepresentedTransformation* to model a transformation of inputs into outputs, which is not the case in the ontological representation of i^* Actor.
  The problem raised up by this observation is fairly similar to the one we encountered in the comparison between i^* Decomposition link, i^* Means ends link and BMM Business Process, that is to say that the similarity Plug-In does not take the created instances and the way they are related into account. Consequently, we don't agree with score obtained and we will validate permanently the initial similarity hypothesis made in step 1. This issue will be discussed in section 10.4.1.

- Concerning the second observation, the ontological mapping of GRL Goal and SoftGoal shows that indeed both instanciate the same ontological concepts as the i^* Actor ontological representation. It is the

reason of the perfect similarity score obtained.
However, the represented properties and represented classes which instanciate those ontological concepts are not related in the same way. There are also more represented properties as well as represented classes in the ontological representation of i^* Actor. Therefore, we can not be fully convinced of the perfect similarity between GRL Goal, GRL SoftGoal and i^* Actor. This problem refers to the one evoqued in the first observation.

Comparison: i^* Goal - GRL constructs

The results of the comparison between i^* Goal and all the GRL constructs are represented in Table 10.8.

GRL constructs	**Results of the comparison with i^* Goal**
SoftGoal	*0,89*
Goal	*0,89*
Resource	*0,67*
Actor	*0,59*
Contribution	*0,57*
Means ends	*0,52*
Decomposition	*0,51*
Dependency	*0,48*
Correlation	*0,48*
Task	*0,46*
Belief	*0,20*

Table 10.8: Results of the comparison between i^* Goal and all the GRL constructs

The analysis of the results depicted in Table 10.8 highlights the high similarity score reaches by GRL Goal (also by GRL SoftGoal). Therefore, the similarity hypothesis we made in step 1 is verified. The similarity score (0,89) exceeds the similarity threshold set at 0,8. Consequently, we will temporaly consider GRL Goal as a highly similar construct in comparison with i^* Goal and therefore temporaly validate the initial similarity hypothesis.

The analysis of the ontology representations of both constructs (see Appendix) confirms the similarity score. Consequently, we will validate permanently the similarity hypothesis made in step 1.

Comparison: *i Task - GRL constructs**

The results of the comparison between *i** Task and all the GRL constructs are represented in Table 10.9.

GRL constructs	Results of the comparison with *i** Task
SoftGoal	*1*
Goal	*1*
Resource	*0,75*
Actor	*0,64*
Contribution	*0,61*
Means ends	*0,56*
Decomposition	*0,56*
Dependency	*0,51*
Correlation	*0,51*
Task	*0,49*
Belief	*0,23*

Table 10.9: Results of the comparison between *i** Task and all the GRL constructs

The results shown in Table 10.9 reveal an observation: they are exactly the same as those depicted in Table 10.7. This noticing underlines indirectly the similarity between the ontological representation of *i** Actor and *i** Task.

Concerning the similarity score reached by GRL Task, it is fairly low (0,49). Therefore, we will temporaly invalidate the similarity hypothesis made in step 1. Now, we will analyse the ontological representations of both constructs in order to gain some knowledge concerning their similarities.

The ontological mappings of *i** Task and GRL Task (see Appendix) reveal different approaches on several aspects in the modelling of the constructs:

- the *i** Task representation models the goal concerned by the task as well as the things concerned by the goal. The GRL Task representation only represents the things concerned by the goal.
- the GRL Task representation decomposes the global task in sub-tasks and specifies some pre-conditions and some post-conditions for each of them. The *i** Task representation adopts a global approach instead. It considers a task as a whole and does neither split it up in sub-tasks nor specifies some conditions on it.
- finally, the GRL Task representation instanciates some represented

states and a represented transformation to model the variation of states of the inputs and outputs concerned by the realization of the goal. The i^* Task representation models the resources needed to perform the task and the things concerned by the goal but it does not represent the states of the inputs and the outputs of the task.

On the basis of those differences, we do agree with the similarity score obtained. Thus, we will invalidate permanently the initial similarity hypothesis made in step 1.

10.4 Conclusion of the comparisons performed

This section has two objectives. The first objective is to assess the tools we used (directly or indirectly) to perform the comparisons; on one hand the similarity Plug-In and on the other hand Protégé. The second objective is to provide a report on the knowledge gained from the comparisons performed.

10.4.1 Assessment of the tools

Assessment of the similarity Plug-In

When comparing two constructs, there are actually two aspects we would like to compare:

1. The first aspect is the representational aspect. We want to know, for each construct, how many represented classes, represented properties, represented states and represented transformations are needed to represent the construct. Moreover, we want to know how they are related between them and what they aim to model. It allows to determine the extent of representational similarity between the two constructs.

2. The second aspect is the semantical aspect. We want to know which ontological concepts the different represented phenomena of each representation instanciate. It allows to determine the extent of semantical similarity between the two constructs.

The second aspect is managed very well by the tool. For each comparison, the similarity Plug-In computes the semantical similarities that exist between the two constructs compared.
Conversely, the first aspect is not managed by the tool. The similarity Plug-In does not take the presentational aspect into account to compute the result of a comparison. The fact that this aspect is not managed by the tool constitutes a hindrance to the completeness of the results obtained.

A possible improvement for the similarity Plug-In would be to implement and to integrate the comparison of representational aspects of constructs into

the functionalities of the tool. The implementation would allow the user to compare some constructs on both aspects separately and therefore to obtain a result for the representational similarity and another one for the semantical similarity. The implementation would also allow the user to recompute the two results in order to obtain a global similarity result. In this way, the completeness of the results obtained would be ensured.

Assessment of Protégé

A suggestion of improvement for Protégé would be to implement the possibility for the user to organize the constructs represented in the common ontology into a hierarchy. It would allow the user to define a specialization relation between certain constructs and therefore to reuse directly the ontological instances created for the "parent" constructs.

This improvement would allow to avoid some problematical situations, such as the one encountered in section 10.2.2 with *i** Agent, *i** Role and *i** Position. Each of those constructs are described as specializations of the construct *i** Actor and are represented in such a way in the ontology. Without any relation of specialization pre-defined between those constructs in the ontology, the similarity Plug-In generated some distort similarity results.
This improvement would also make certain language analysis easier, especially those concerning the languages which contain a lot of specialized concepts.

10.4.2 Knowledge gained from the comparisons

Having performed several comparisons in section 10.2 and 10.3, we are now able to evaluate the results produced by the tool on the basis of two different EML approaches.
When considering only the semantical aspect of the constructs compared and when taking into account the threshold set, the results obtained during the comparisons revealed the following observations:

- when comparing a Business oriented EML (BMM) with a Goal oriented EML (*i**), four comparisons out of six refuted the intial similarity hyptohesis made on the constructs; in other words nearly 67% of refutation.

- when comparing a Goal oriented EML (*i**) with another Goal oriented EML (GRL), two comparisons out of three refuted the intial similarity hyptohesis made on the constructs; also nearly 67% of refutation.

These observations lead to conclude that hypothetically similar constructs taken from EMLs that have an identical EM approach (*i** and GRL for

instance) won't necesseraly share more semantical commonalities than hypothetically similar constructs taken from EMLs that have different EM approaches (BMM and i^*). The semantical similarities will rather depend on how the language analyst perceive, interpret and translate the constructs of a language into ontological concepts.

10.5 Summary

This chapter tested out a similarity Plug-In in order to analyse the relevance of the information bring out by this tool. It also highlighted its limitations. First, we defined a comparison methodology in order to support a certain sort of comparison performed by the tool. Then, we applied this methodology throughout the comparisons performed thanks to the similarity Plug-In. We faced several problematical situations when analysing the comparison results brought out by the tool. It allowed us to emphasize some limitations of the tool. Then, we proposed some suggestions to improve the similarity Plug-In and Protégé. Finally, we provided a report on the knowledge gained from the comparisons performed.

Part IV

Conclusion

Chapter 11

Conclusion

11.1 The problem

Enterprise modelling (EM) is used by companies from various branches of industry. It allows those companies to model several enterprise matters such as organization, resources, information, requirements, goals and strategy. This broad scope of application gave rise to many different enterprise modelling languages (EMLs). Many EM technologies available on the market offered efficient but different functionalities and semantics. Therefore, they were rarely able to interoperate and to communicate. The resulting enterprise models were often incompatible and they prohibited the interoperability of working solutions. A real need for EM interoperability appeared. The idea of a Unified Enterprise Modelling Language (UEML) emerged in order to bring a solution to these numerous EMLs problems.
In the first part of the thesis, we expounded the purposes pursued by the two versions of UEML. We briefly introduced the first version (UEML 1.0), and we then explored in details the approach defined by the second version (UEML 2.0). This approach describes a specific methodology to incorporate EMLs in UEML. analysing and incorporting new EMLs in UEML contributes to the very essence of UEML: the more diverse are the languages integrated in UEML and the richer and the more efficient becomes UEML as an interoperability solution.

11.2 Contribution

This thesis contributes to analyse two EMLs, BMM and i^*, using the UEML 2.0 approach in order to incorporate them in UEML.
BMM is a Business oriented model whereas i^* is a Goal oriented framework. Therefore, those two EMLs have different purposes. On one hand, BMM offers a structure by way of which business plans can be developed, communicated and managed in an organized manner. On the other hand, the

i^* framework proposes two models: a Strategic Dependency Model for describing the network of relationships among actors and a Strategic Rationale Model for modelling the reasoning that each actor has about its relationships toward other actors.
Chapter 7 applied the UEML 2.0 approach to a selected construct of BMM and chapter 8, in turn, focused on analysing a construct of i^* EML in order to integrate it into UEML.

Prior to those two chapters, chapter 6 described two important methdologies set up in this thesis and used within the context of the UELM 2.0 approach. The first methodology plays a part in the construct description phase of the UEML 2.0 and allows to clear up ambiguities which appear in constructs descriptions. The second methodology, on the basis of a decomposition of the constructs descriptions in terms of some represented phenomena, aims at mapping the constucts onto a common ontology and is thus used in the mapping phase of UEML 2.0. The reason to set up these methodologies was to support the UEML approach.

Following the contribution, the third part of the thesis mainly consists of an evaluation of a similarity tool and had two goals. The first goal was to improve the languages analysis realised previously by using a tool called UEML Validator. The improvement areas found were however minor. The second goal of the evaluation was to try out a comparison tool, the Similarity Plug-In, on the constructs of BMM, i^* and GRL. The Plug-In allows to detect similarities between these languages constructs. Although the Plug-in could be improve, mainly on the representational aspect, the knoweldge we gained from the tool was that hypothetically similar constructs taken from EMLs that have an identical EM approach will not necesseraly share more semantical commonalities than hypothetically similar constructs taken from EMLs that have different EM approaches.

11.3 Future works

In order to enable UEML to become an efficient interoperability solution for entreprises modelling needs, the next logical step of the project should be the consolidation of the results obtained from the different languages analysis. Because there is a part of subjectivity in the analysis work, the different specalists who carried on the analysis have to reach a global agreement on the languages analysis (constructs descriptions and mappings).

Another track of development for UEML would be to used a tool such as an improved similarity Plug-In to detect commonalities of different languages constructs. These commonanlities could then serve as basis to define the next

version UEML, which would not be a collection of constructs from different languages anymore but become an EML on its own.

Bibliography

[BAO04] Giuseppe Berio, Victor Anaya and Angel Ortiz.*Supporting Enterprise Integration through a Unified Enterprise Modeling Language.* 2004.

[Ber03] Giuseppe Berio.*Requirements analysis : initial core constructs and architecture.* May 2003

[Ber05a] Giuseppe Berio. *UEML 2.0 Deliverable 5.1 - INTEROP project.* 2005.

[Ber05b] Giuseppe Berio. *UEML 1.0 and UEML 2.0: Benefits, Problems and Comparison.* 2005.

[BOAD05] Giuseppe Berio, Andreas Opdahl, Victor Anaya, Michele Dassisti. *Deliverable DEM1 - UEML 2.1.* 2005.

[BO06] Giuseppe Berio, Andreas opdahl. *Interoperable Language and Model Management using the UEML Approach.* 2006.

[BPP04] Giuseppe Berio, Herve Panetto, Michaël Petit. *UEML: résultats et enjeux d'un langage unifié de modélisation d'entreprise.* September 2004.

[BRG05] BRG. *The Business Motivation Model - Business Governance in a Volatile World.* September 2005.

[Bun77] Mario Bunge. *Treatise on Basic Philosophy, Volume 3, chapter Ontology I: The Furniture of the World.* Reidel, Boston, 1977.

[Bun79] Mario Bunge. *Treatise on Basic Philosophy, Volume 4, chapter Ontology II: A World of Systems.* Reidel, Boston, 1979.

[DES05] DES. *Organization Structure Metamodel (OSM) - Draft Submission Overview.* December 2005.

[Dou02] Guy Doumeingts.*UEML (Unified Enterprise Modelling Language).* 2002

[GYV06] Jaap Gordijn, Eric Yu, and Bas van der Raadt. *e-Service Design Using i* and e3value Modeling.* 2006.

[Ko03] John Krogstie. *Evaluating UML using a generic quality framework.* 2003.

[Mah06] Jeremy Mahiat. *A Validation tool for the UEML Approach.* 2006.

[MWU] *Merriam-Webster Unabridged Dictionary.*

[OMG06] OMG. *Business Motivation Model (BMM) Specification.* June 2006.

[Op06] Andreas opdahl, *The UEML Approach to Modelling Construct Description.*March 2006

[OXF] *Oxford Dictionary of English.*

[Petit] Michaël Petit, Guy Doumeingts. *Report on the State of the Art in Enterprise Modelling.* August 2002.

[SBVBRS06] OMG.*Semantics of Business Vocabulary and Business Rules Specification.* March 2006.

[Shane] David Shane. *Youth must learn skills to succeed in Global Economy*

[SI03] Shariff, Ismail. *global economic integration: prospects and problems. From An International Journal of Development Economics. Development Review, Vol1, No.2.* 2003.

[Ver02] François Vernadat.*UEML: towards a unifed enterprise modelling language.* 2002

[WW88] Yair Wand and Ron Weber. *An ontological analysis of some fundamental information systems concepts. Ninth International Conference on Information Systems*, 30 December 1988.

[WW93] Yair Wand and Ron Weber. *On the ontological expressiveness of information systems analysis and design grammars. Journal of Information Systems*, 1993.

[WW95] Yair Wand and Ron Weber. *On the deep structure of informations systems. Journal of Information Systems*, 1995.

[Yu95] Eric Yu. *Modelling Strategic Relationships for Process Reengineering.* 2006.

Appendix A

BMM Mappings

A.1 BMM assessment

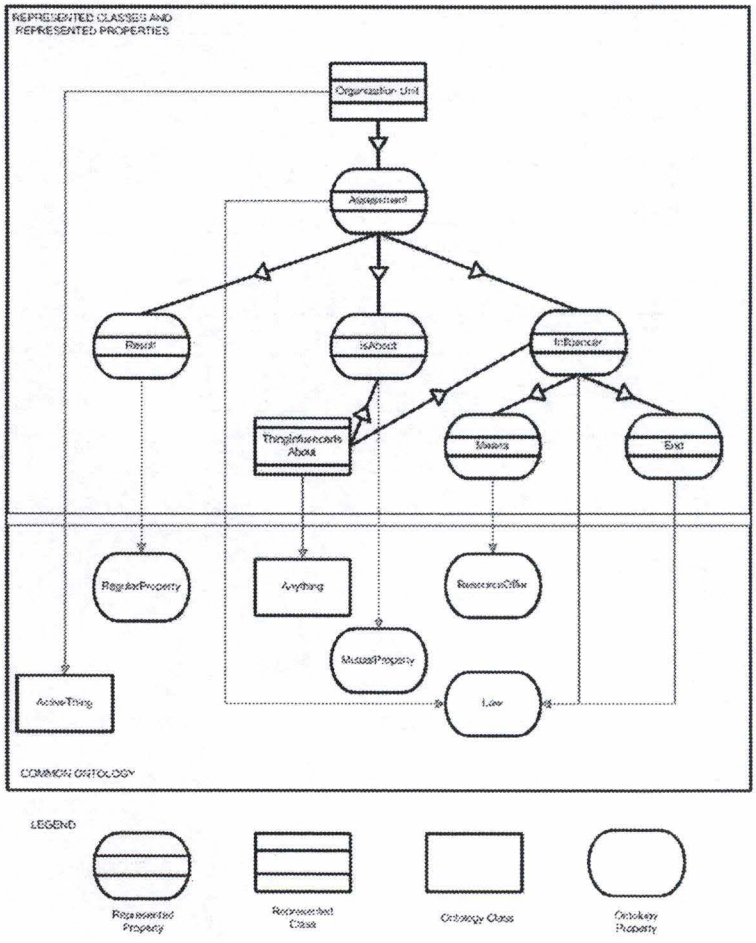

Figure A.1: Ontological mapping of BMM assessment

A.2 BMM assumption

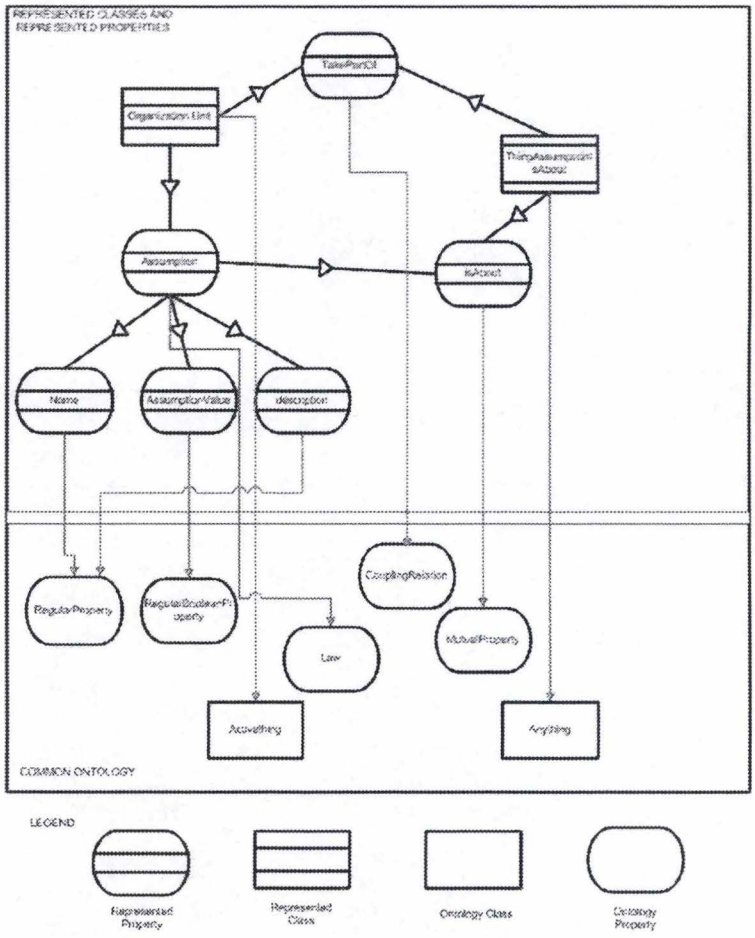

Figure A.2: Ontological mapping of BMM assumption

A.3 BMM Business Policy

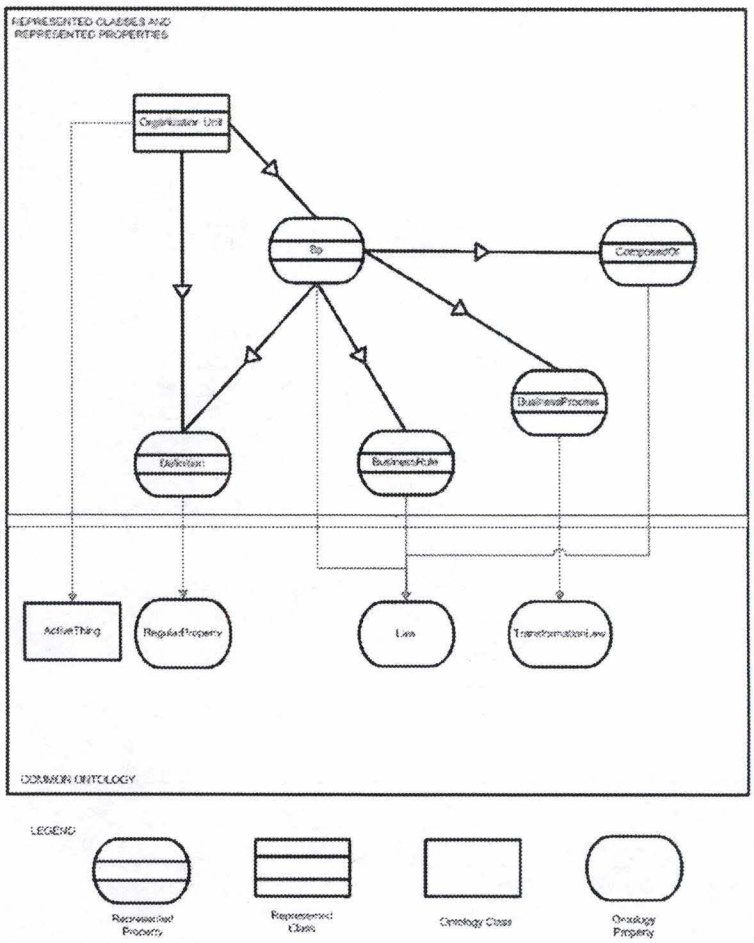

Figure A.3: Ontological mapping of Business Policy

A.4 BMM Business Process

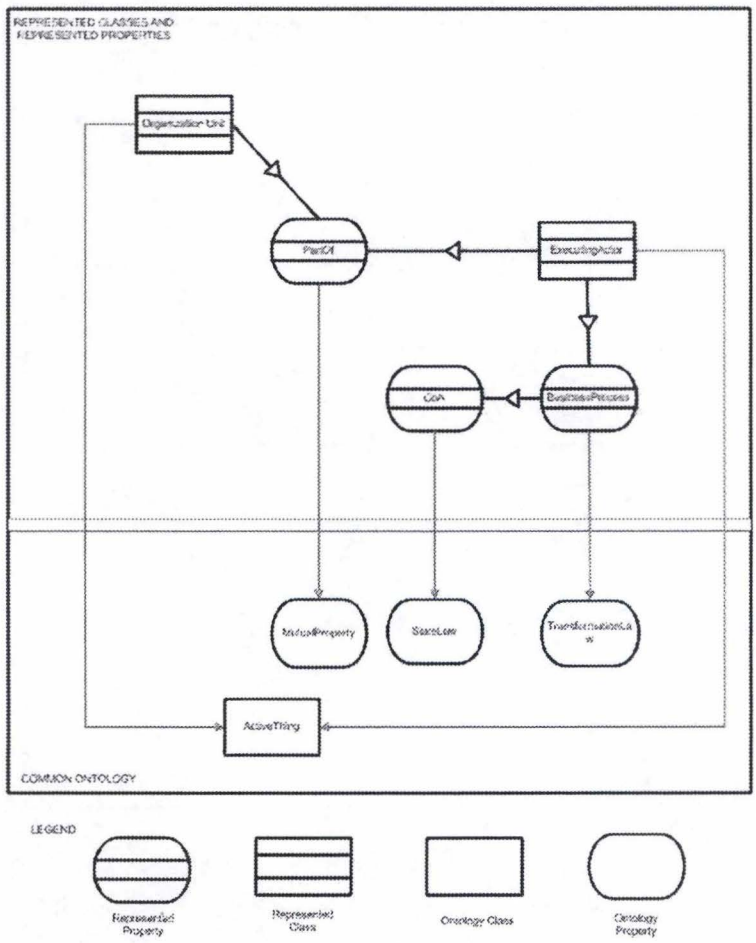

Figure A.4: Ontological mapping of BMM Business Process

A.5 BMM Business Rule

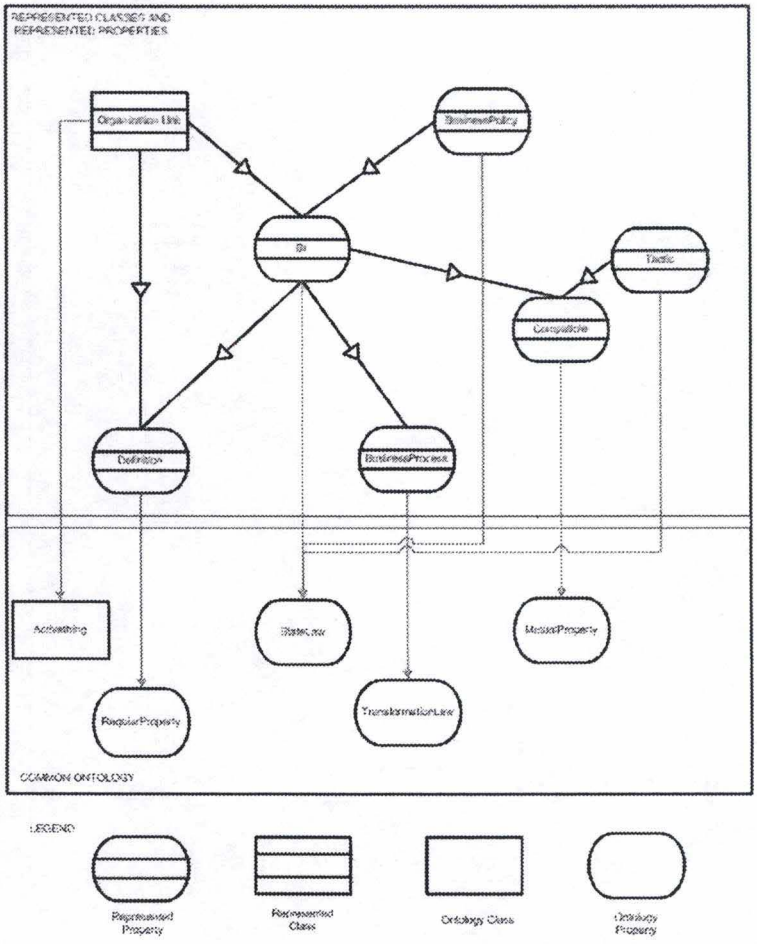

Figure A.5: Ontological mapping of BMM Business Rule

A.6 BMM Competitor

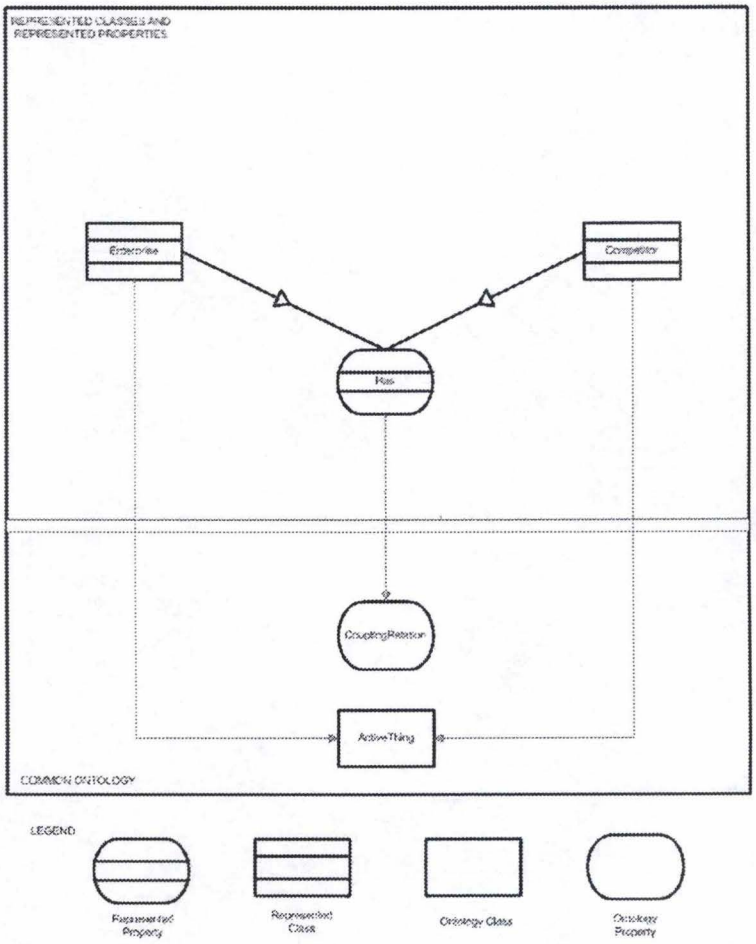

Figure A.6: Ontological mapping of BMM Competitor

A.7 BMM Corporate Value

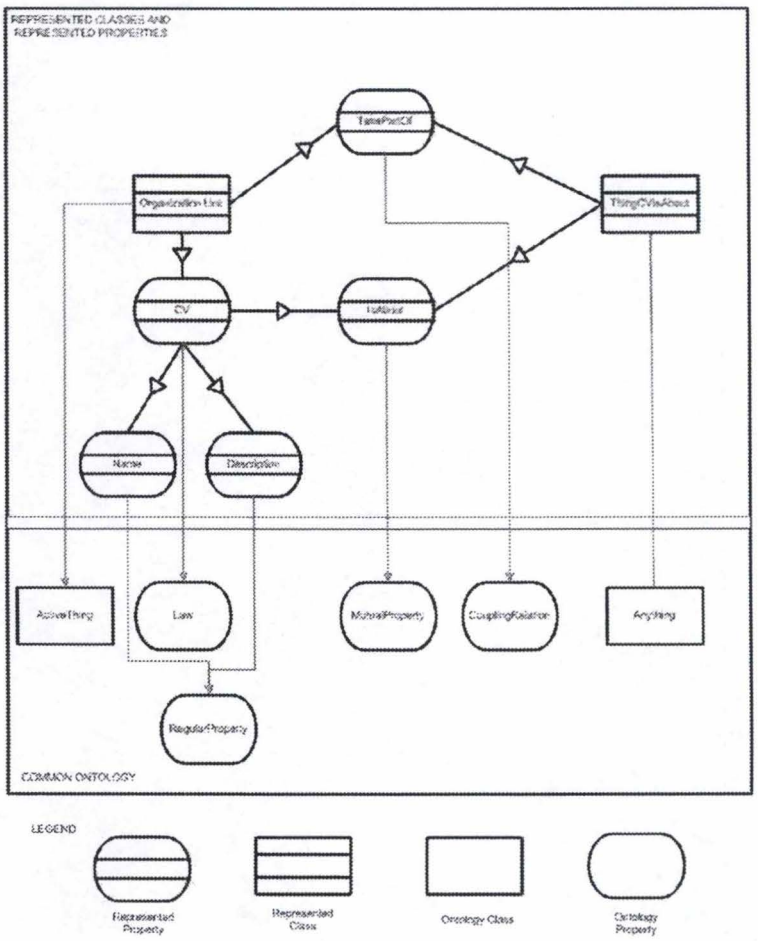

Figure A.7: Ontological mapping of BMM Corporate Value

A.8 BMM Course Of Action

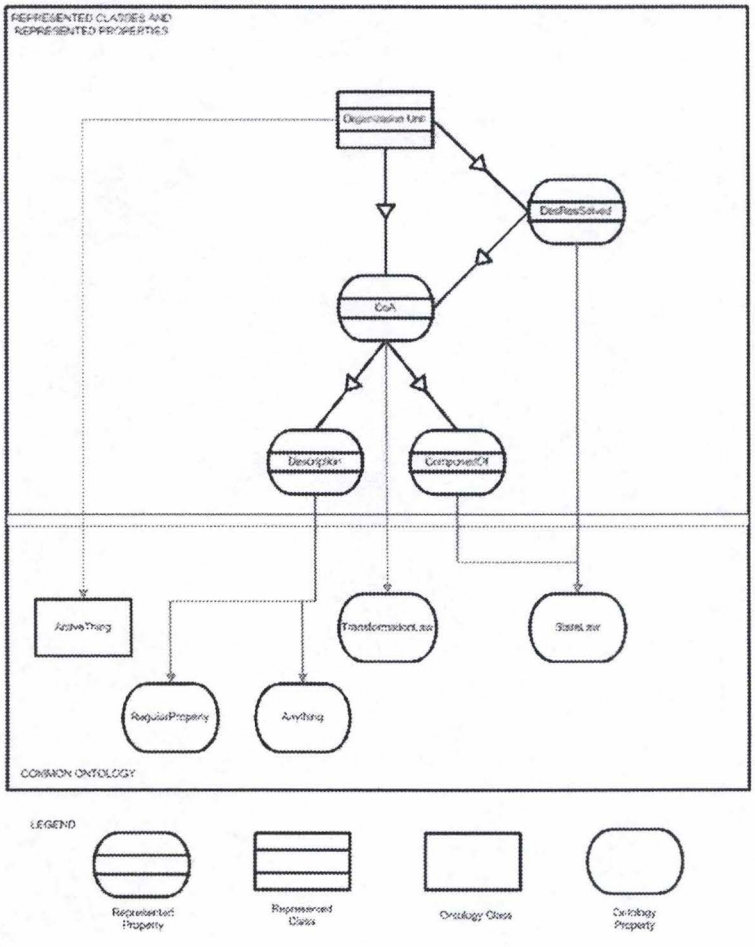

Figure A.8: Ontological mapping of BMM Course Of Action

A.9 BMM Customer

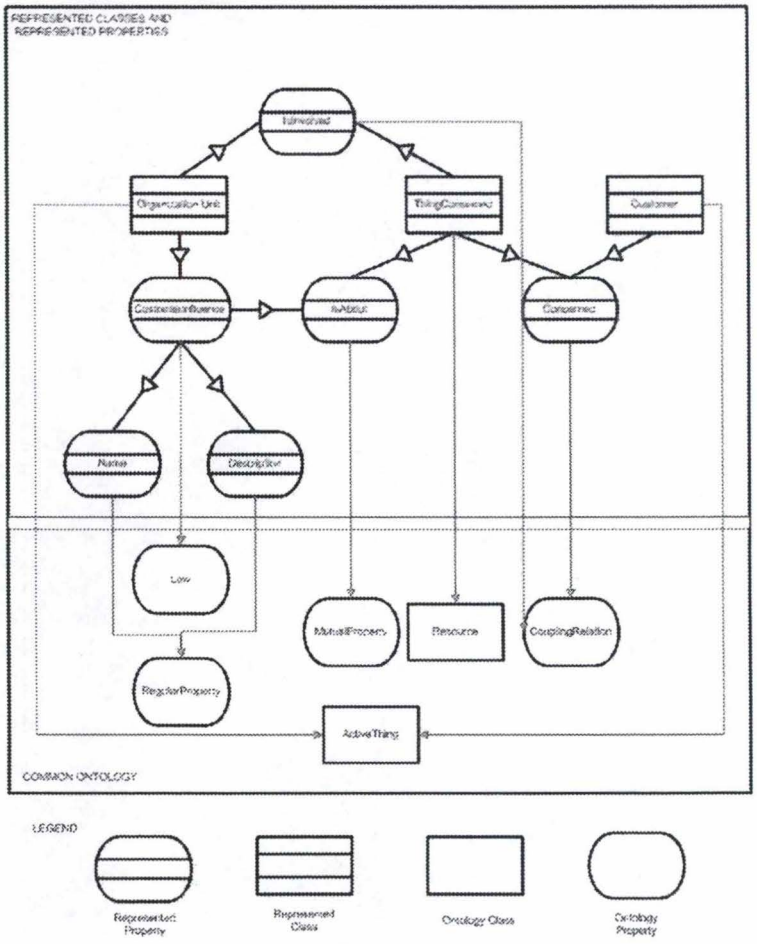

Figure A.9: Ontological mapping of BMM Customer

A.10 BMM Desired Result

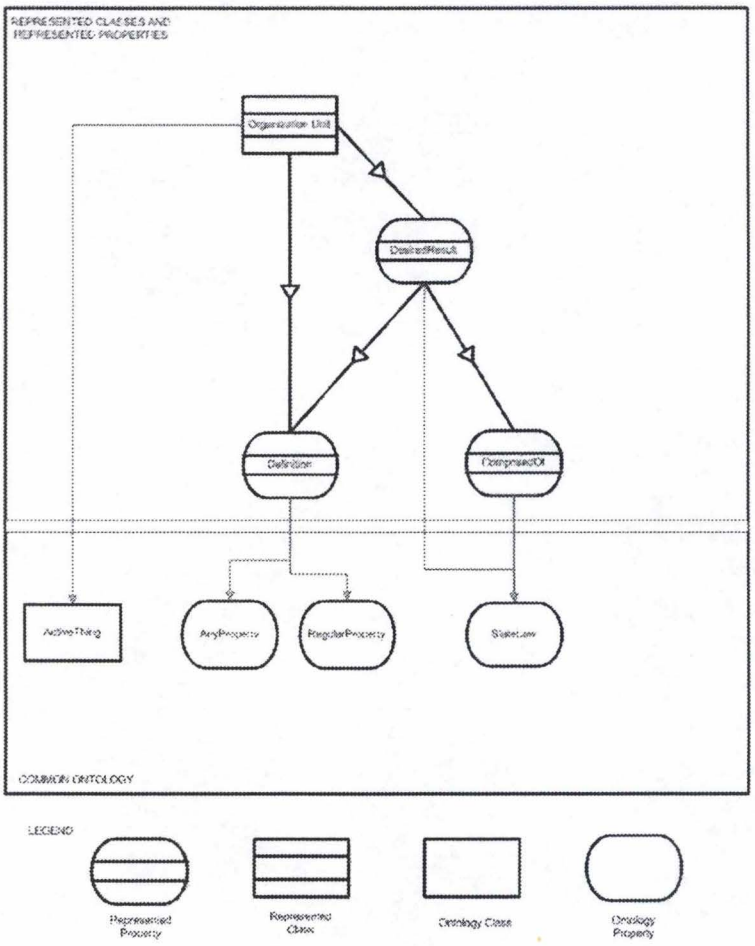

Figure A.10: Ontological mapping of BMM Desired Result

A.11 BMM Directive

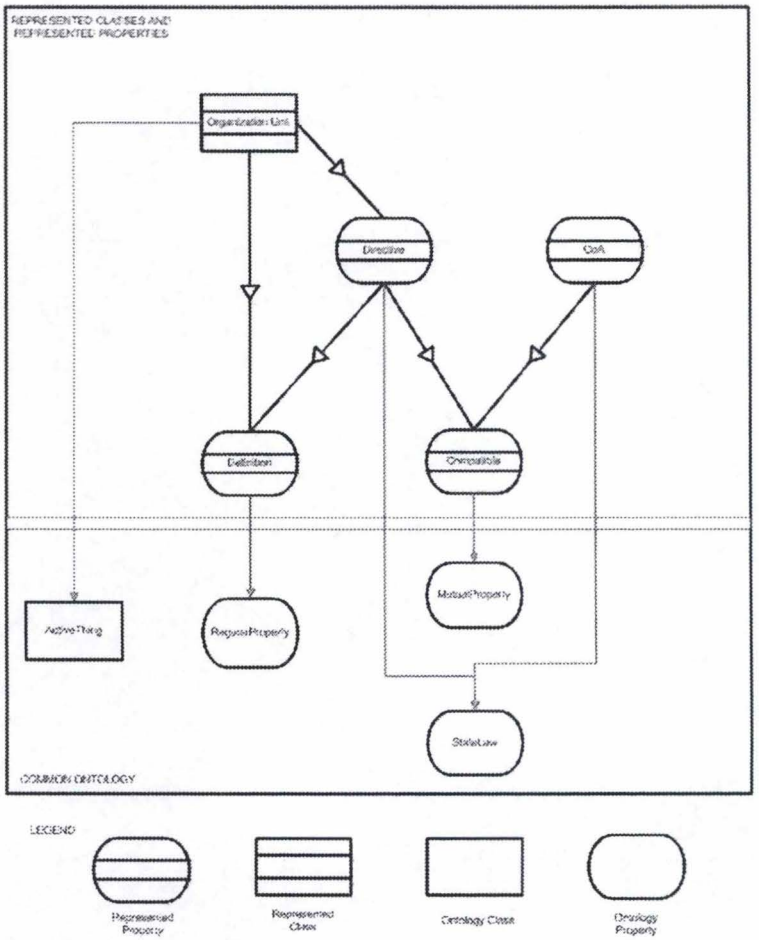

Figure A.11: Ontological mapping of BMM Directive

A.12 BMM End

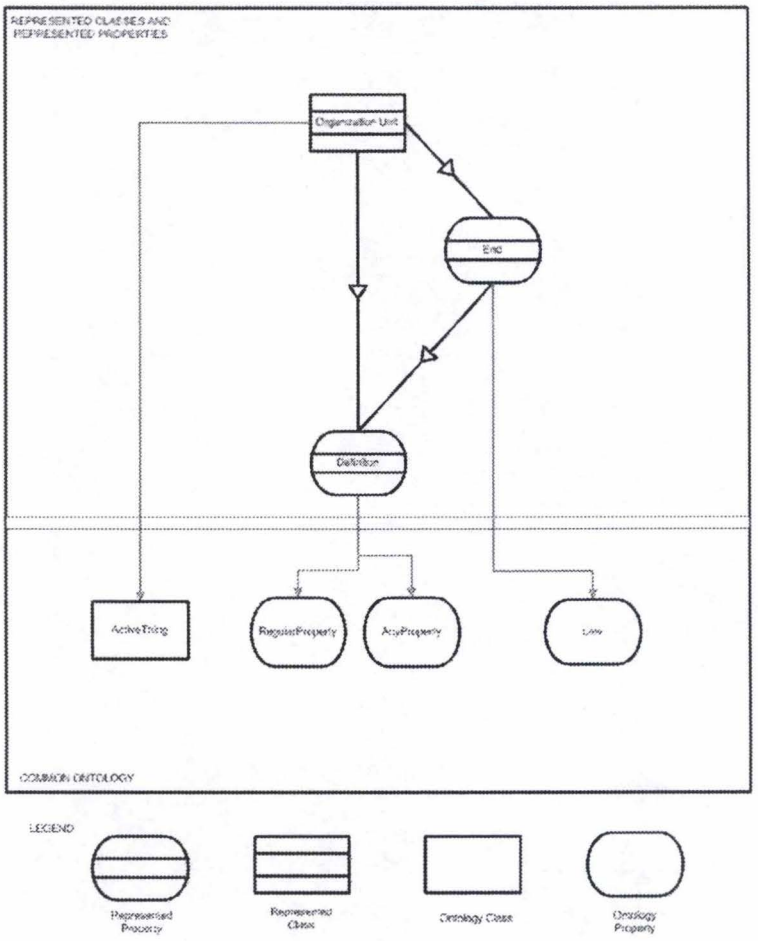

Figure A.12: Ontological mapping of BMM End

A.13 BMM Environment

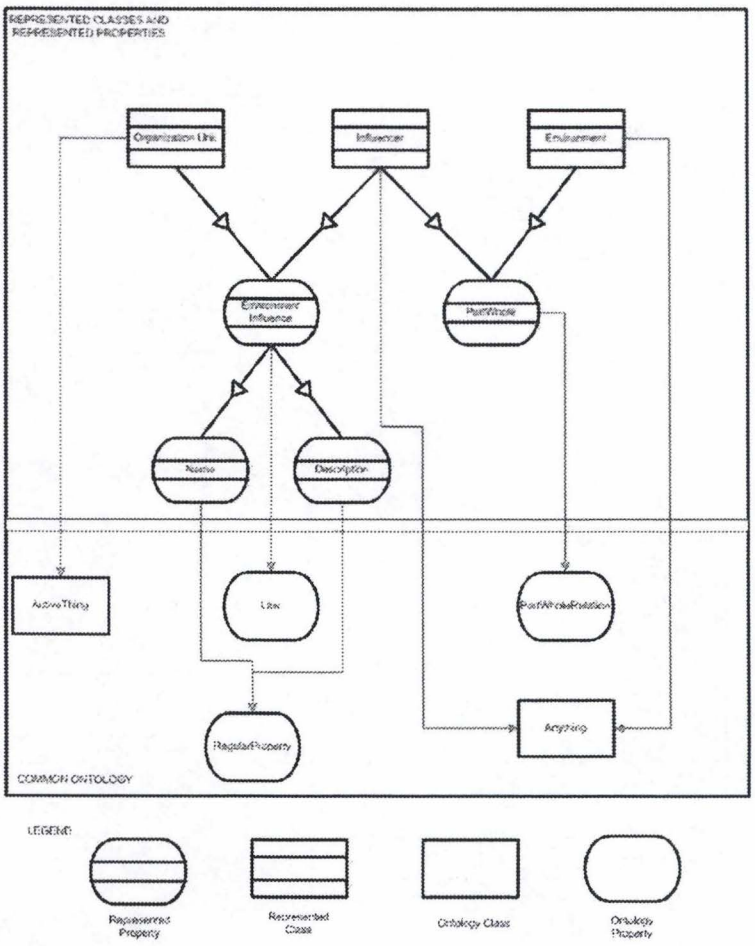

Figure A.13: Ontological mapping of BMM Environment

A.14 BMM Explicit Corporate Value

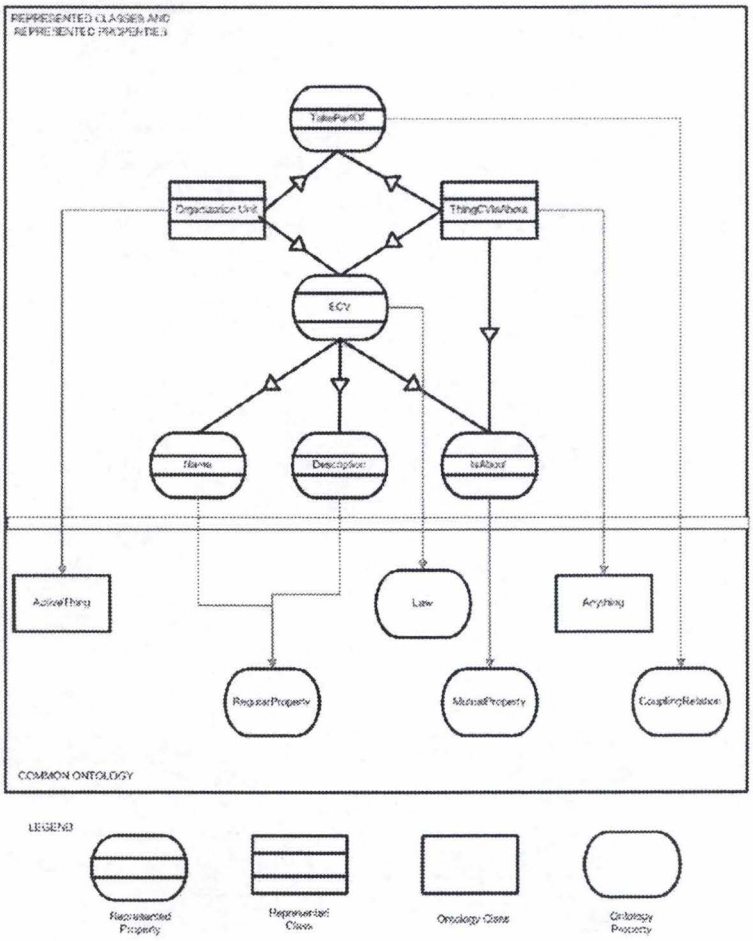

Figure A.14: Ontological mapping of BMM Explicit Corporate Value

A.15 BMM External Inflencer

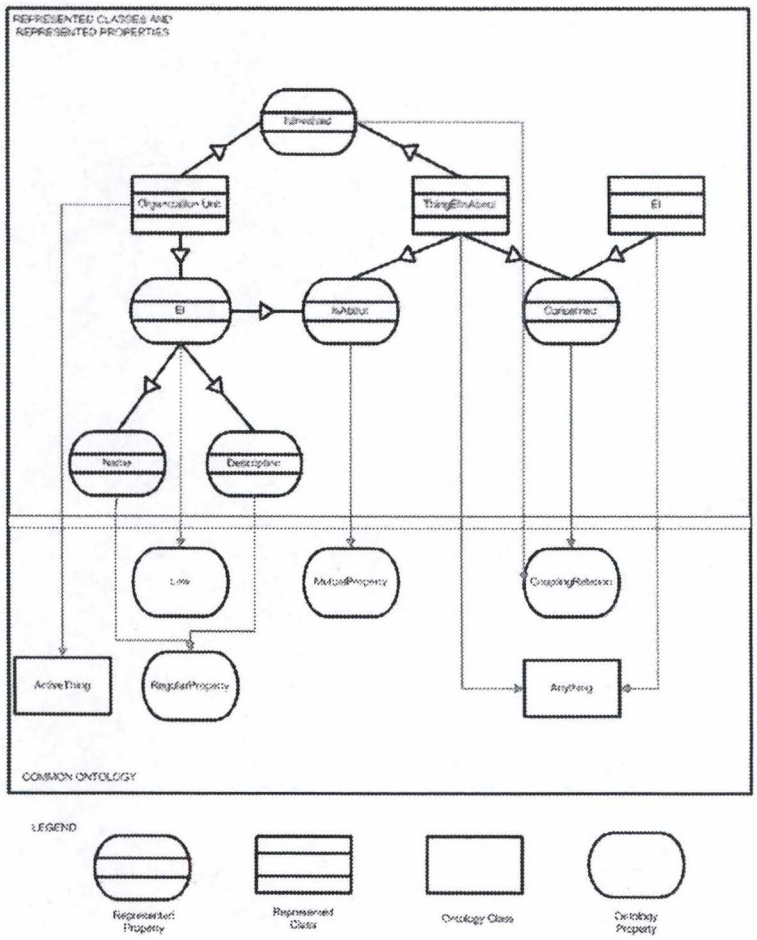

Figure A.15: Ontological mapping of BMM External Inflencer

A.16 BMM Goal

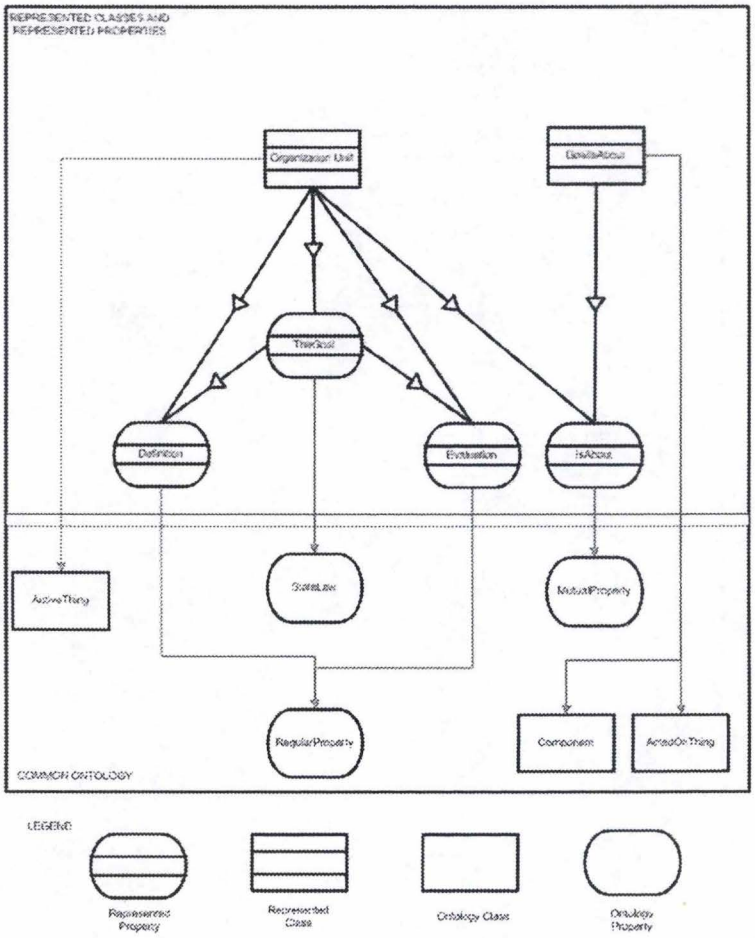

Figure A.16: Ontological mapping of BMM Goal

A.17 BMM Habit

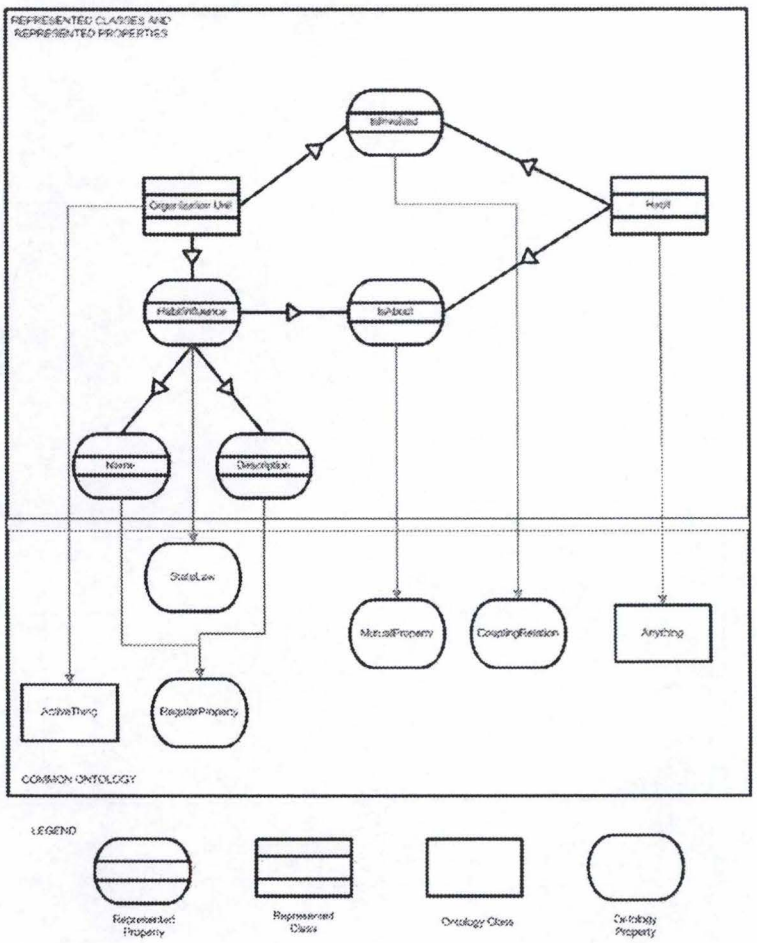

Figure A.17: Ontological mapping of BMM Habit

A.18 BMM Implicit Corporate Value

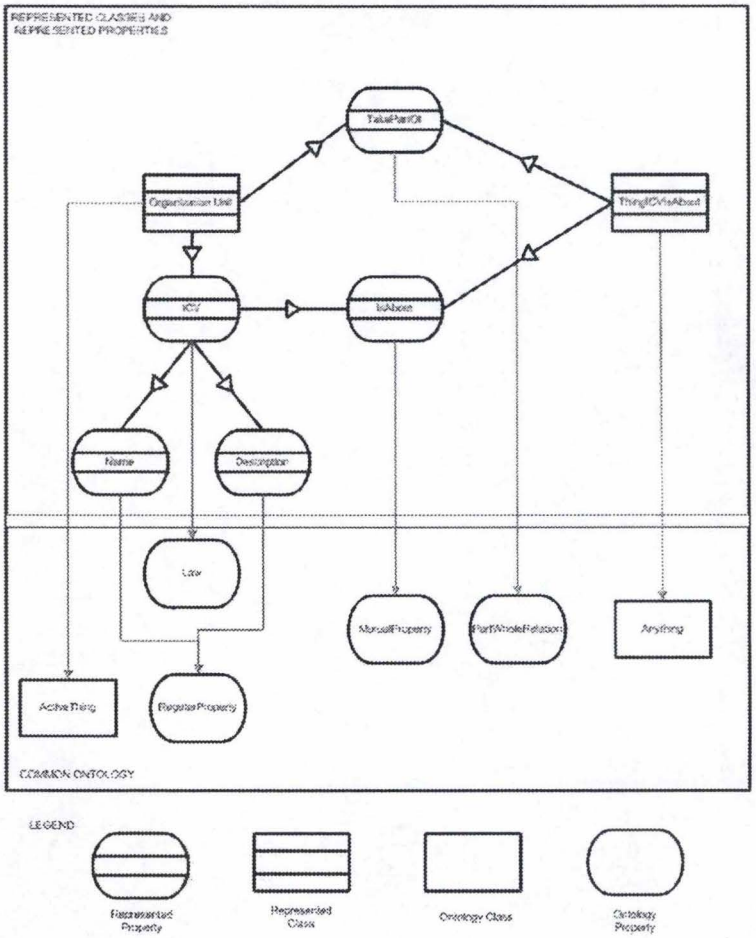

Figure A.18: Ontological mapping of BMM Implicit Corporate Value

A.19 BMM Influencer

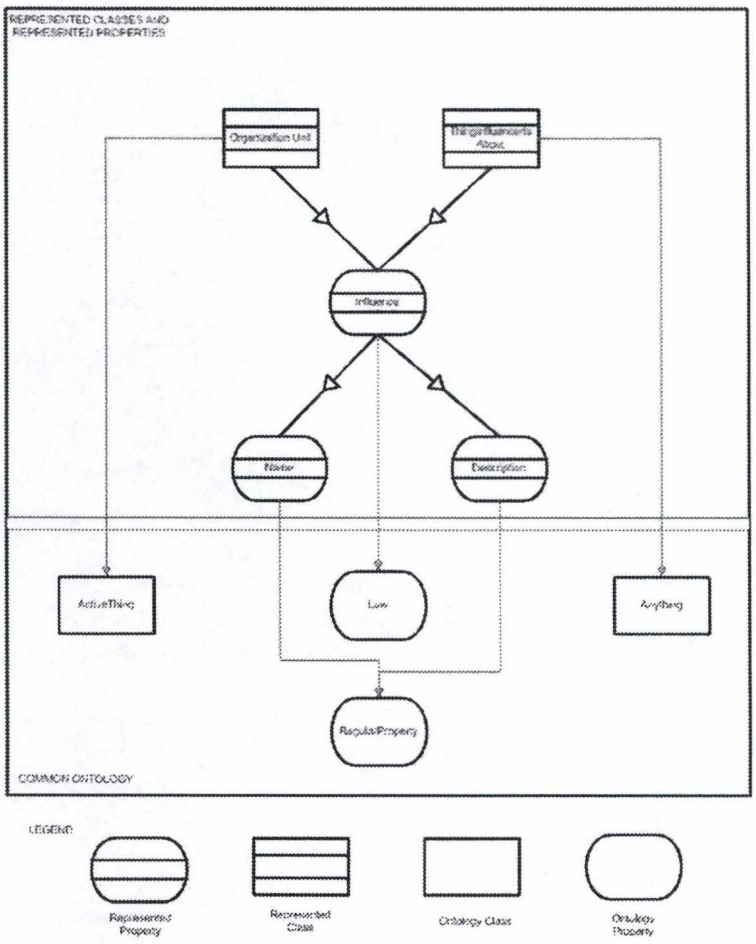

Figure A.19: Ontological mapping of BMM Influencer

A.20 BMM Infrastructure

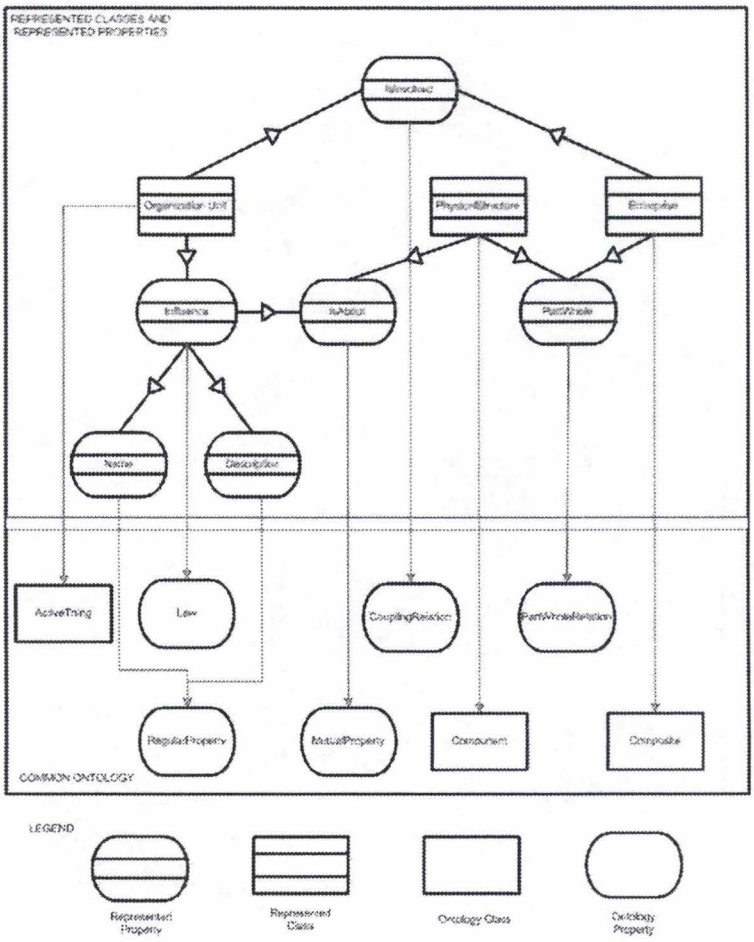

Figure A.20: Ontological mapping of BMM Infrastructure

A.21 BMM Internal Influencer

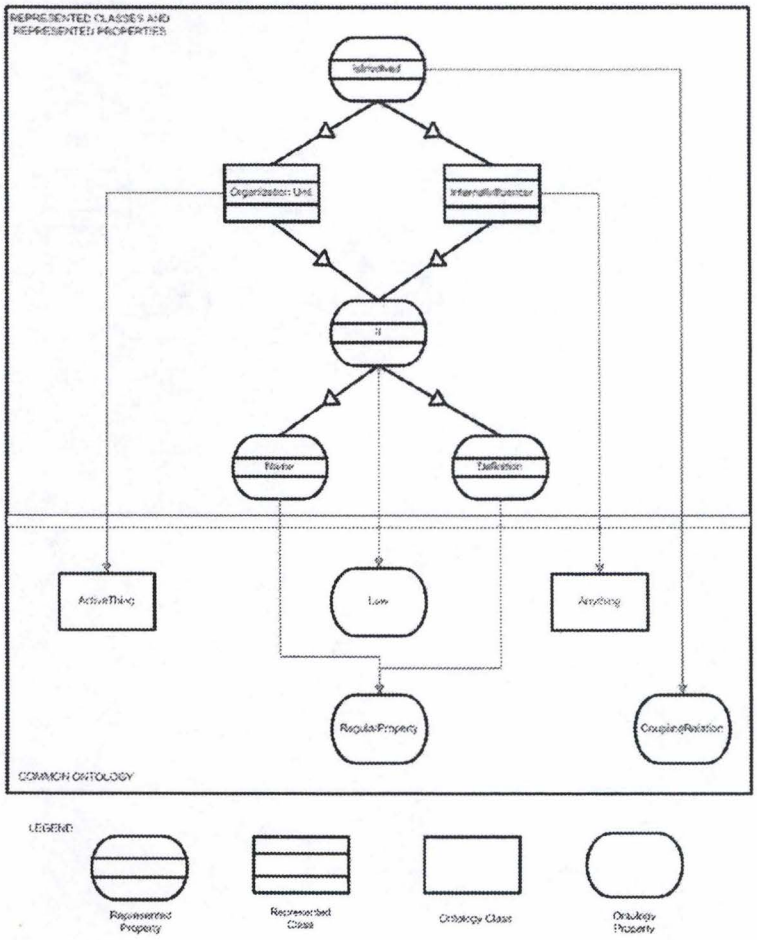

Figure A.21: Ontological mapping of BMM Internal Influencer

A.22 BMM Issue

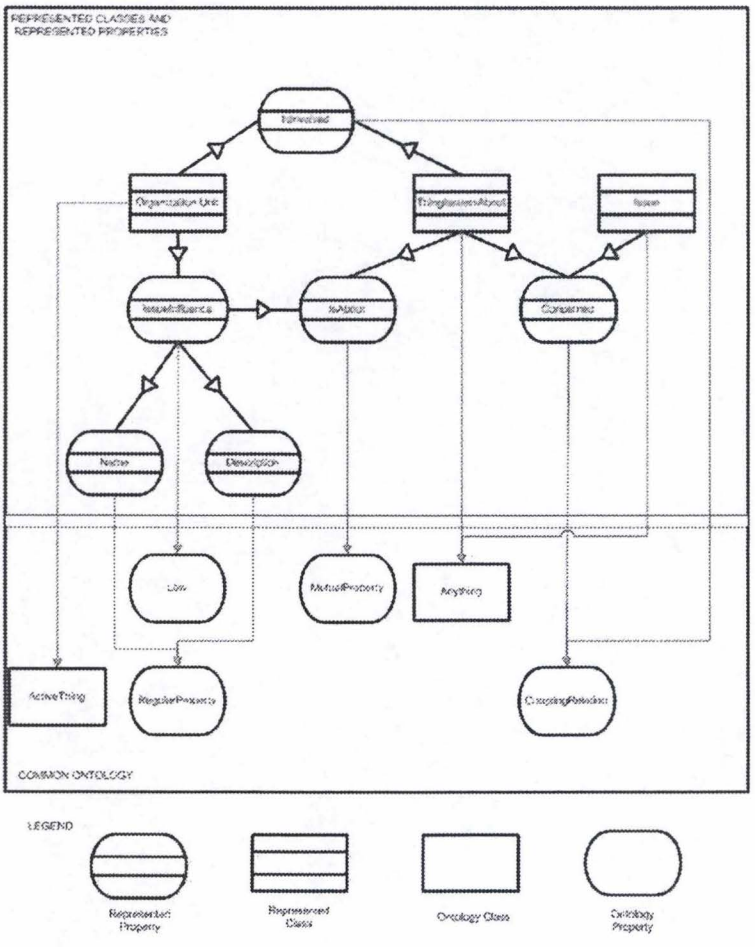

Figure A.22: Ontological mapping of BMM Issue

A.23 BMM Management Prerogative

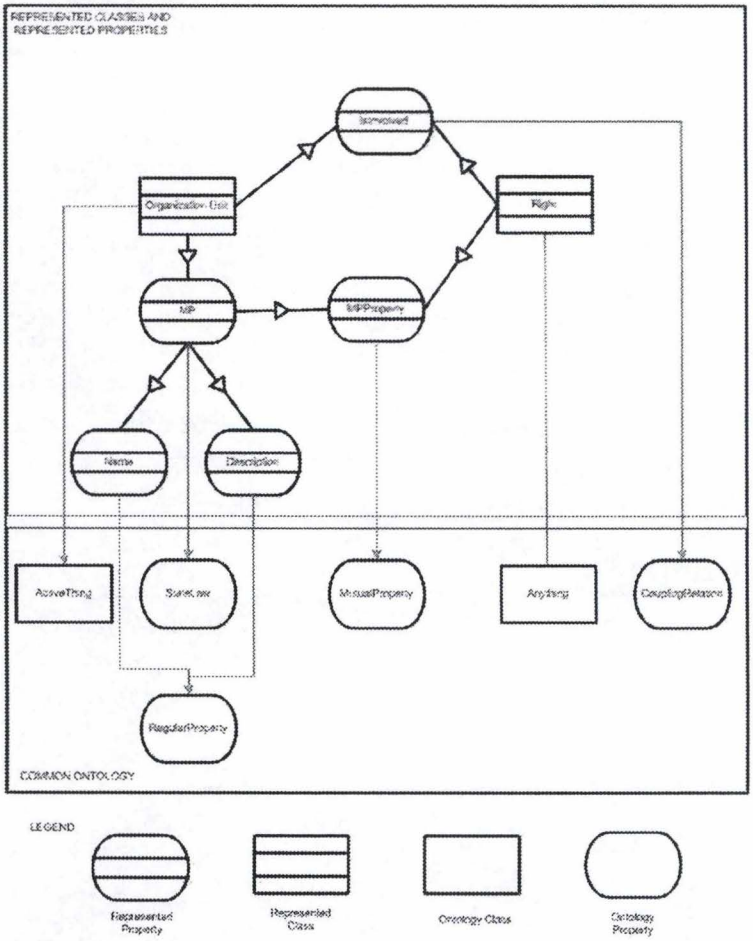

Figure A.23: Ontological mapping of BMM Management Prerogative

A.24 BMM Means

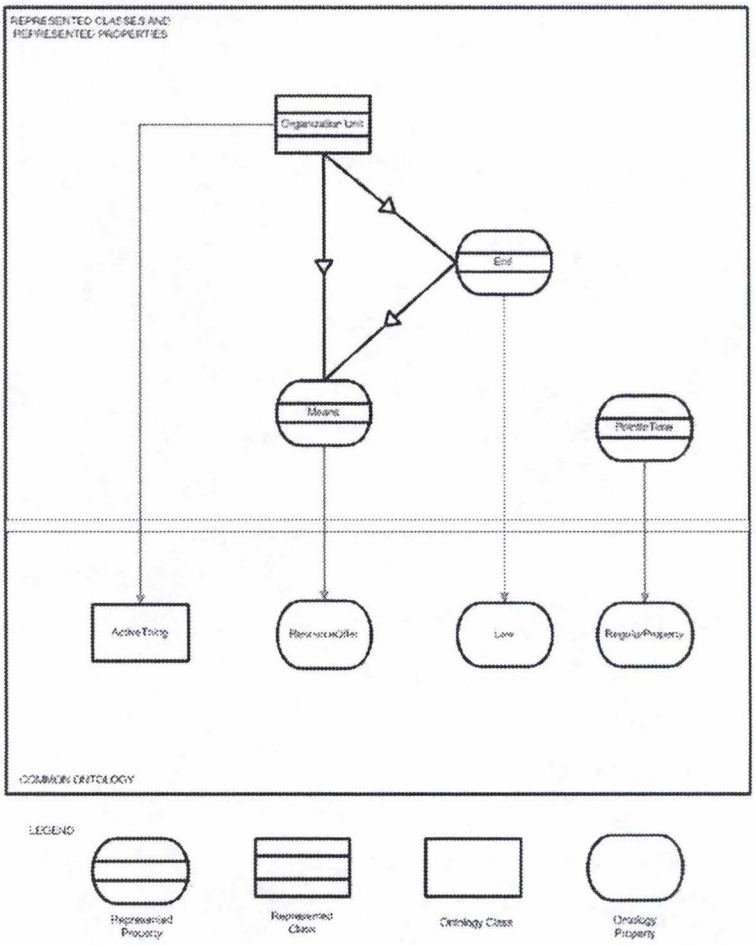

Figure A.24: Ontological mapping of BMM Means

A.25 BMM Mission

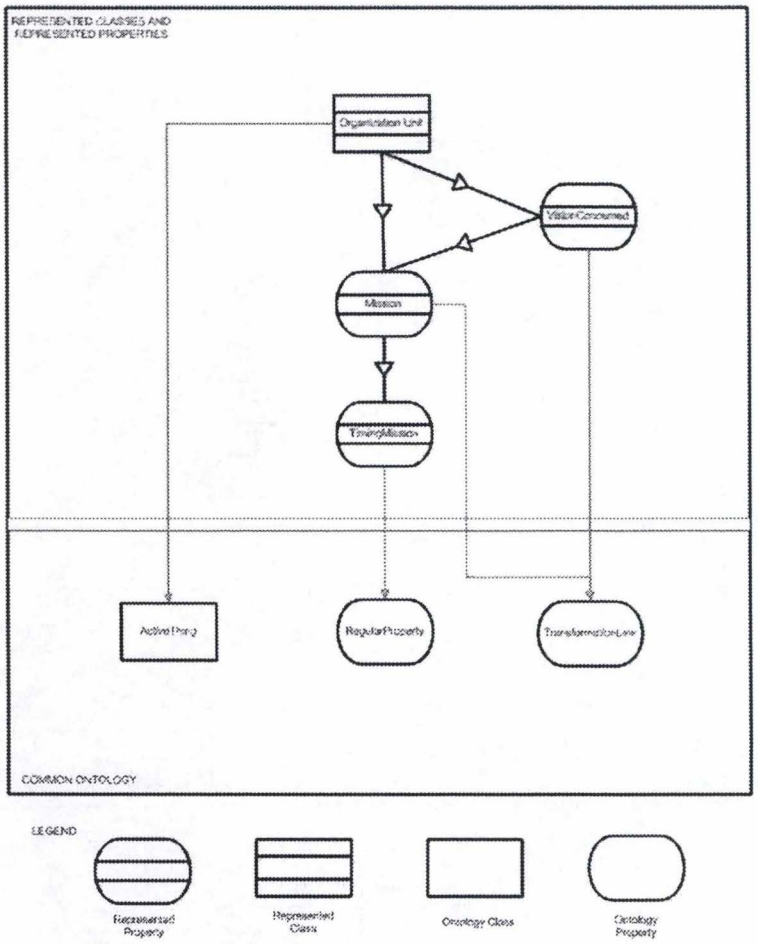

Figure A.25: Ontological mapping of BMM Mission

A.26 BMM Objective

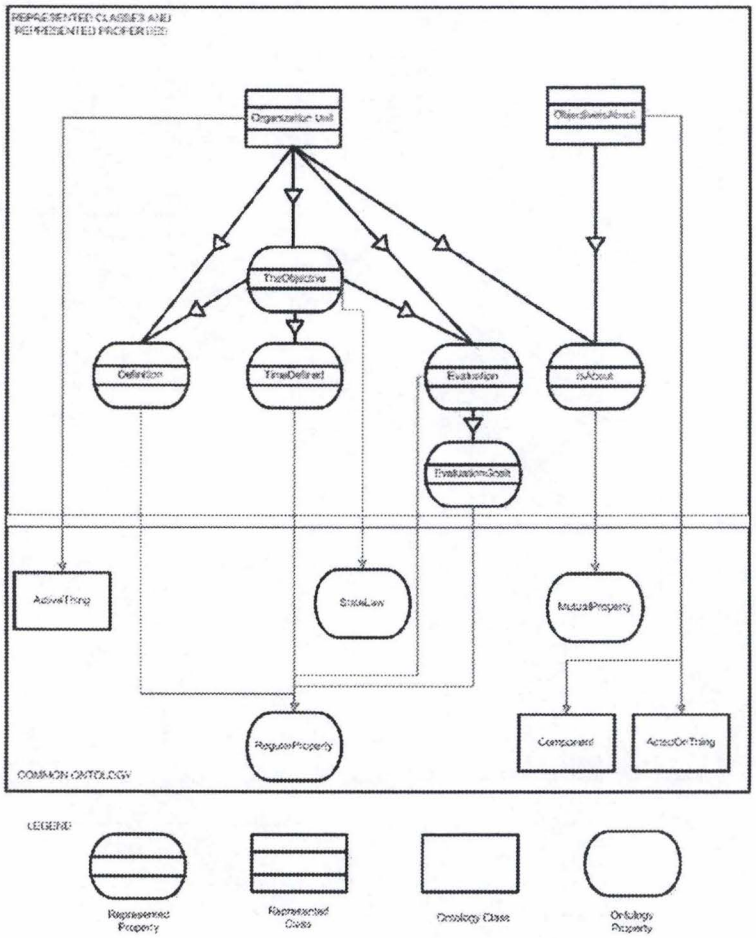

Figure A.26: Ontological mapping of BMM Objective

A.27 BMM Opportunity

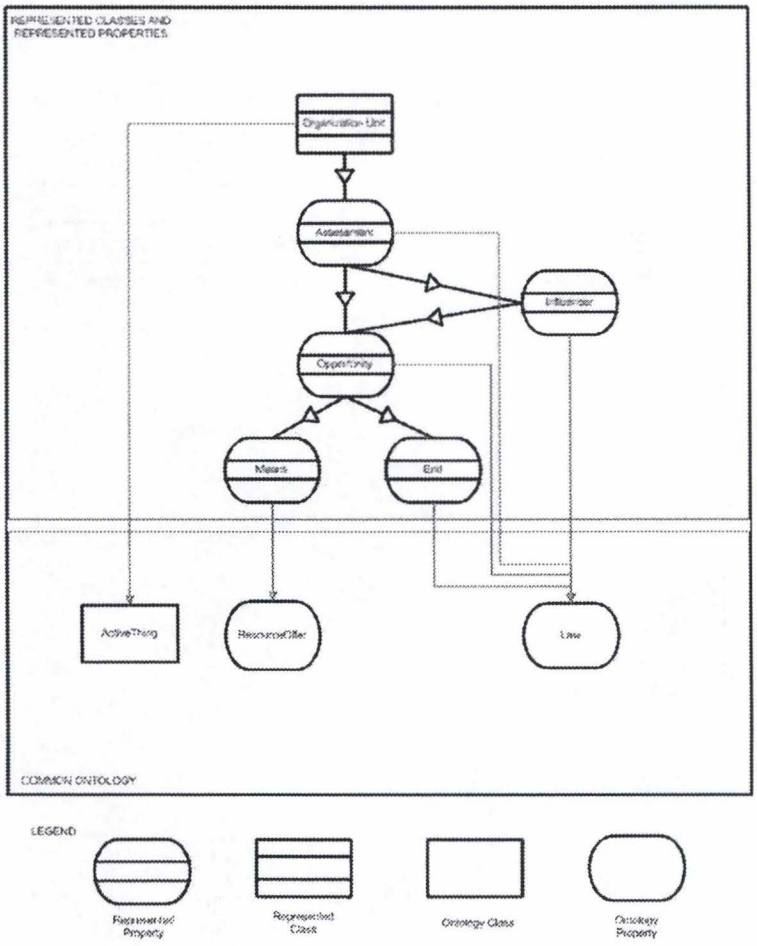

Figure A.27: Ontological mapping of BMM Opportunity

A.28 BMM Partner

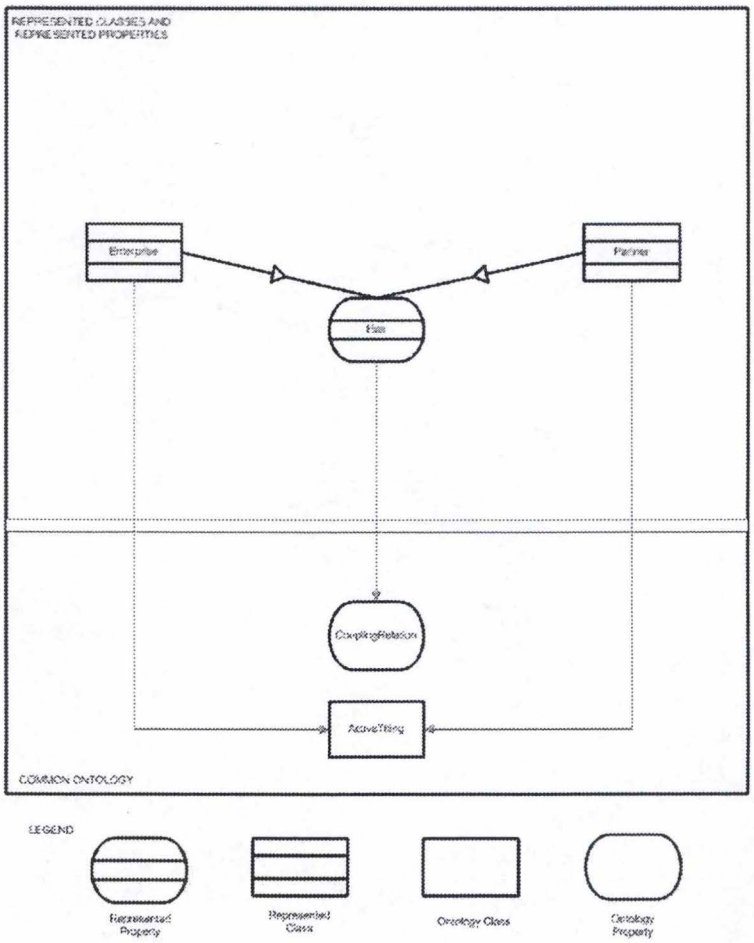

Figure A.28: Ontological mapping of BMM Partner

A.29 BMM Opportunity

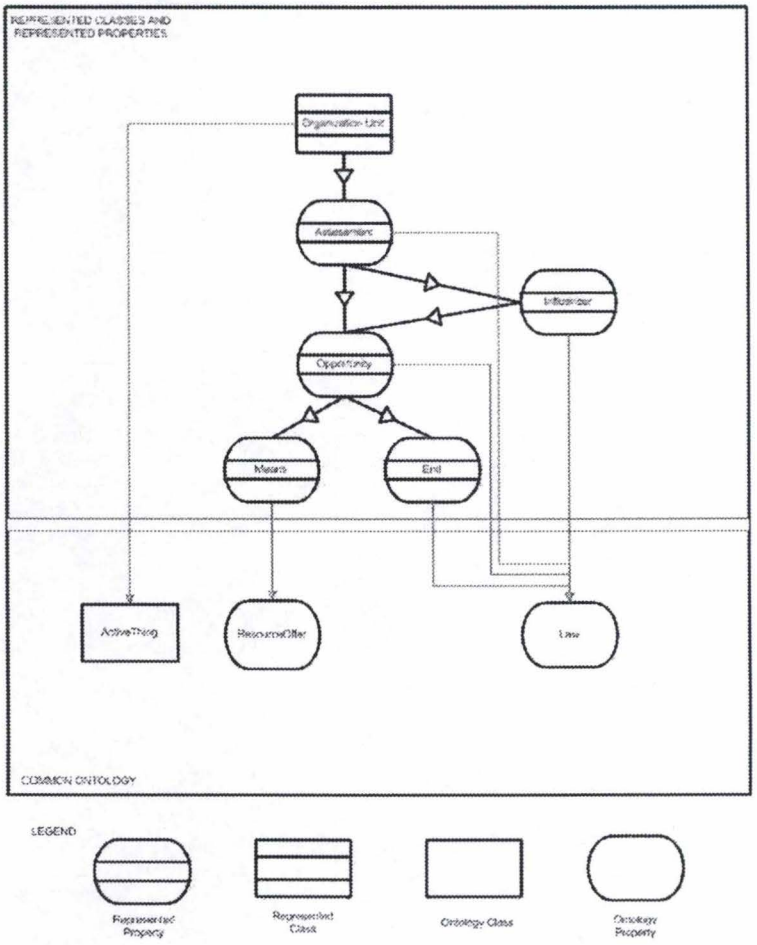

Figure A.29: Ontological mapping of BMM Opportunity

A.30 BMM Potential Award

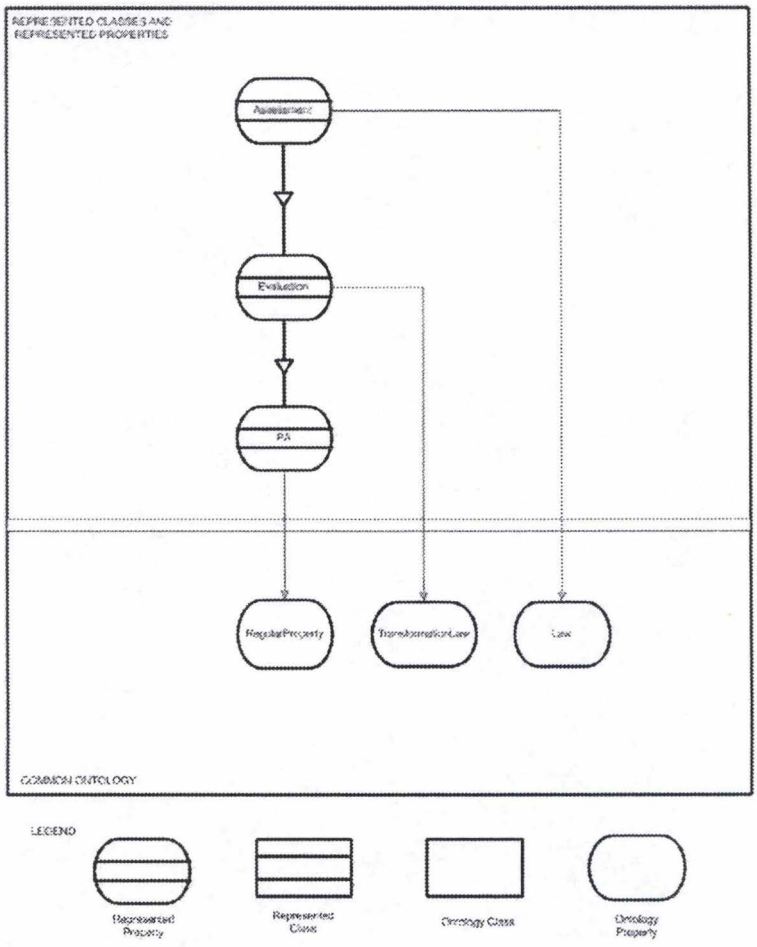

Figure A.30: Ontological mapping of BMM Opportunity

A.31 BMM Regulation

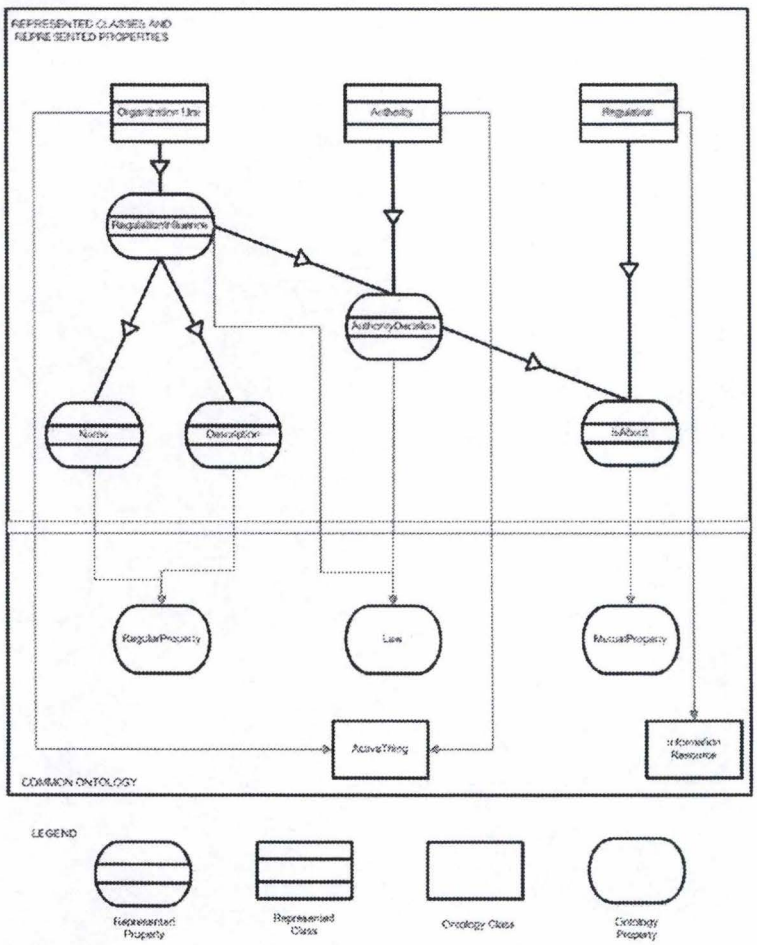

Figure A.31: Ontological mapping of BMM Regulation

A.32 BMM Potential Resource

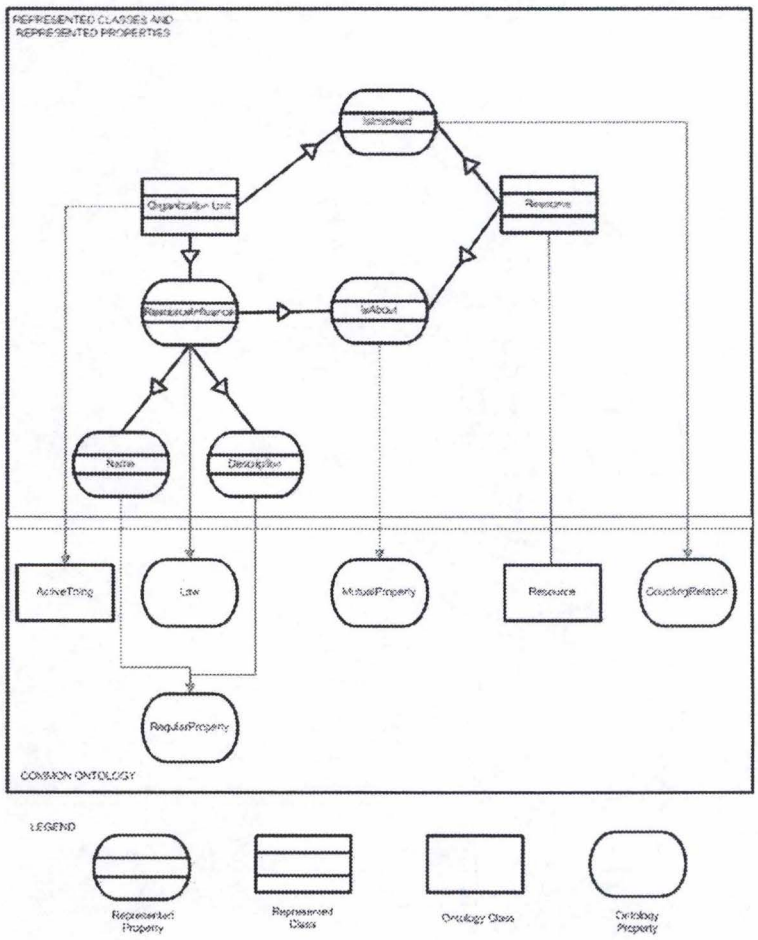

Figure A.32: Ontological mapping of BMM Resource

A.33 BMM Risk

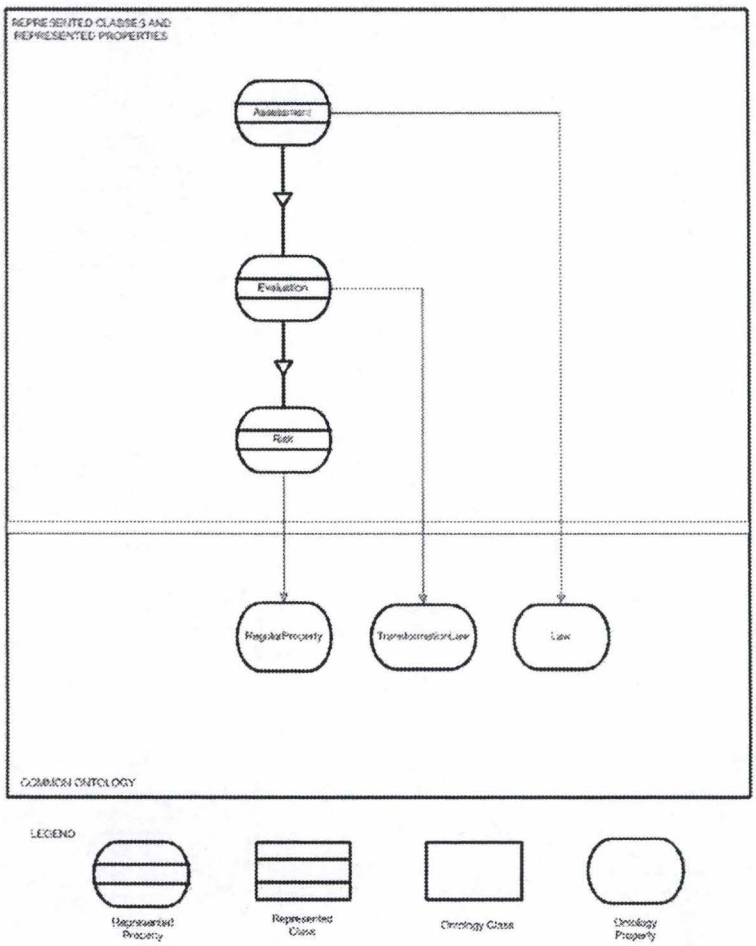

Figure A.33: Ontological mapping of BMM Risk

A.34 BMM Strategy

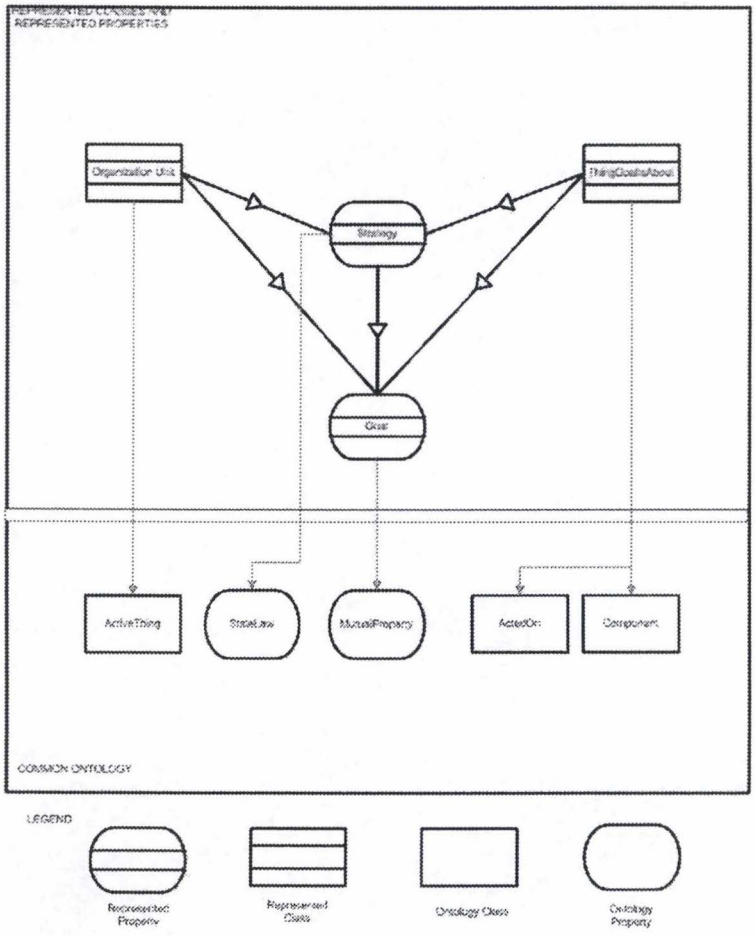

Figure A.34: Ontological mapping of BMM Strategy

A.35 BMM Strength

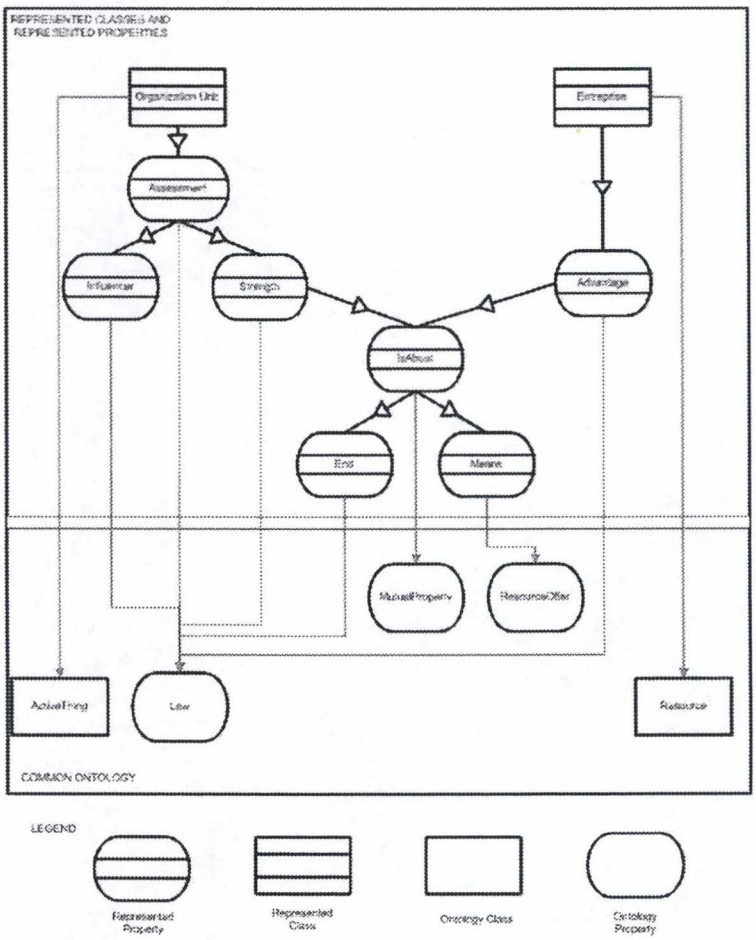

Figure A.35: Ontological mapping of BMM Strength

A.36 BMM Supplier

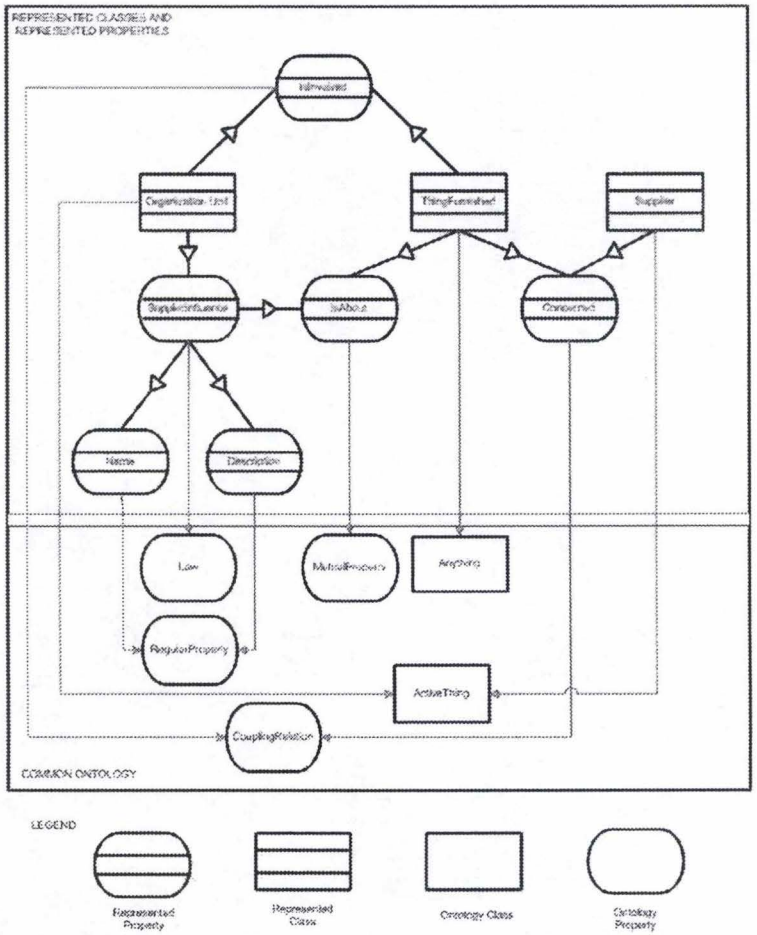

Figure A.36: Ontological mapping of BMM Supplier

A.37 BMM Tactic

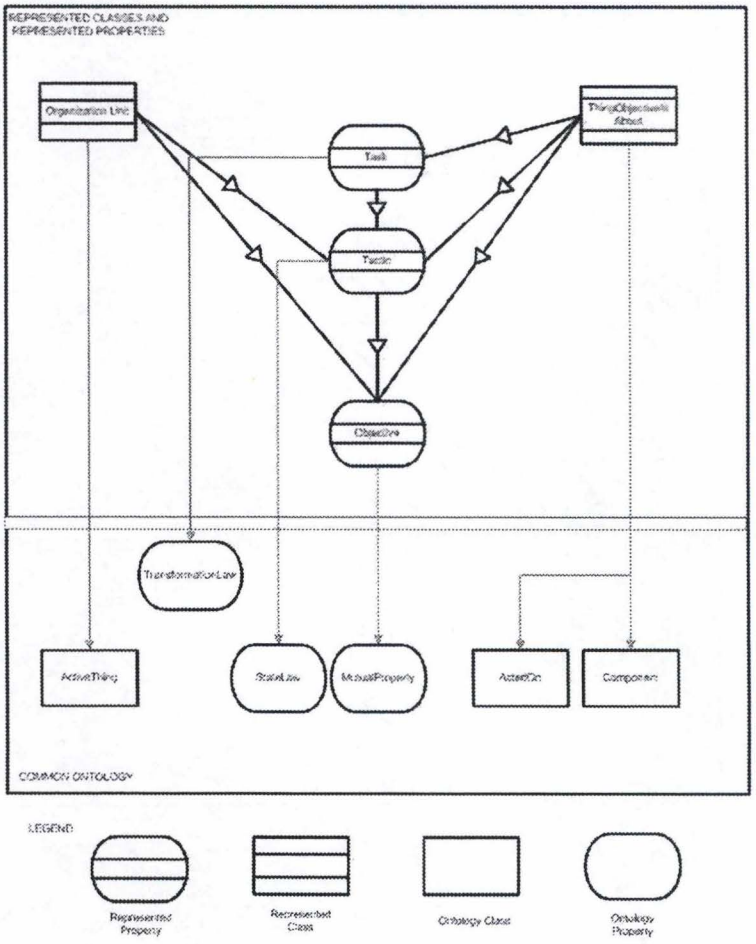

Figure A.37: Ontological mapping of BMM Tactic

A.38 BMM Technology

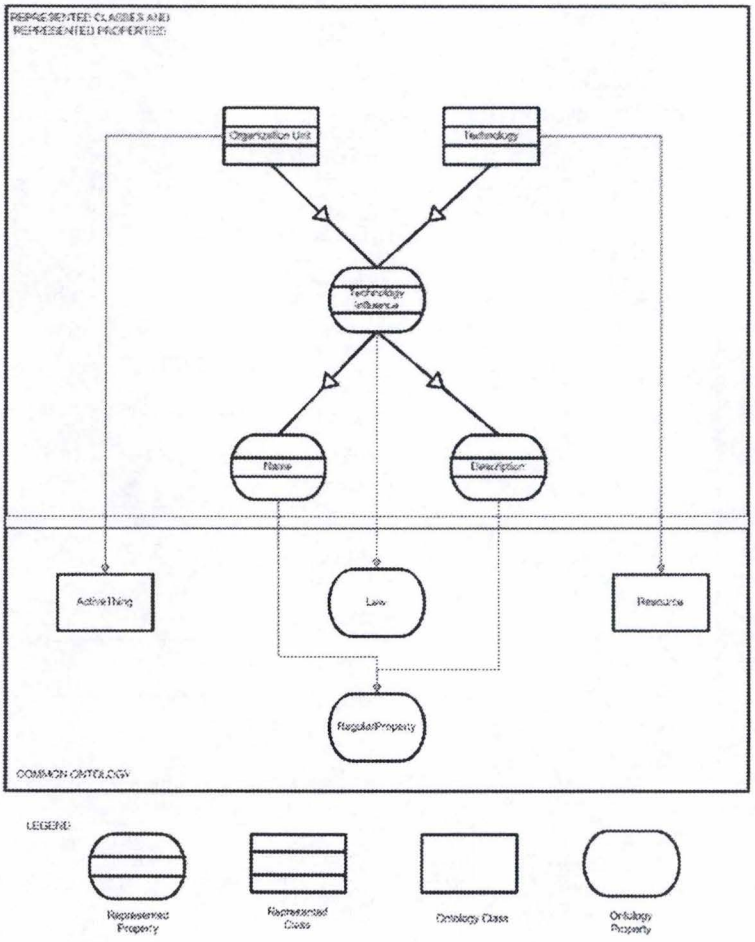

Figure A.38: Ontological mapping of BMM Technology

A.39 BMM Threat

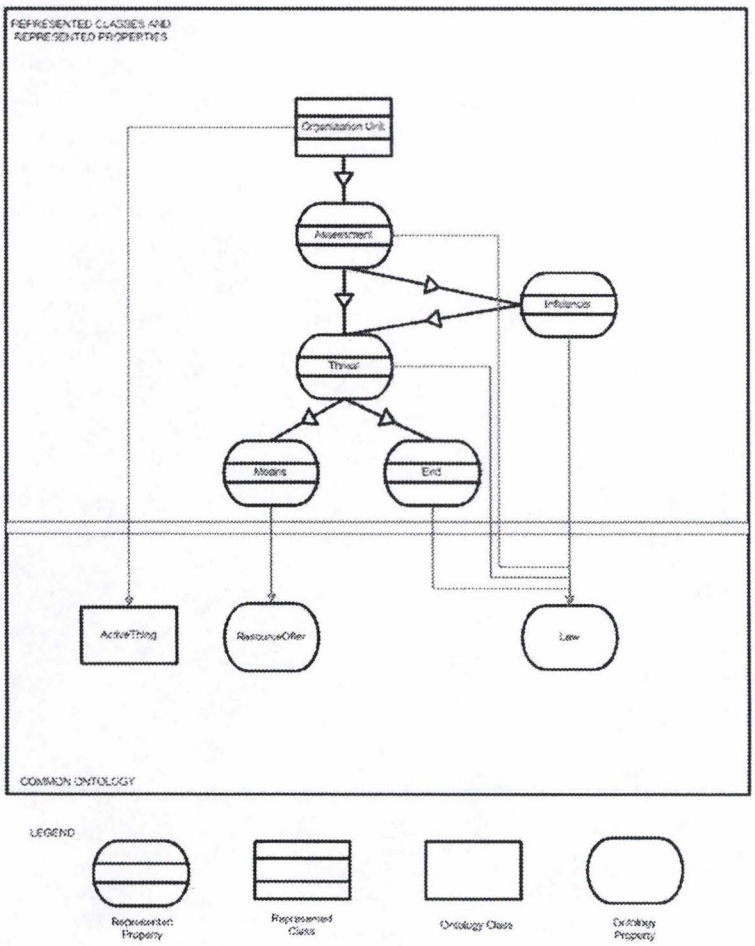

Figure A.39: Ontological mapping of BMM Threat

A.40 BMM Vision

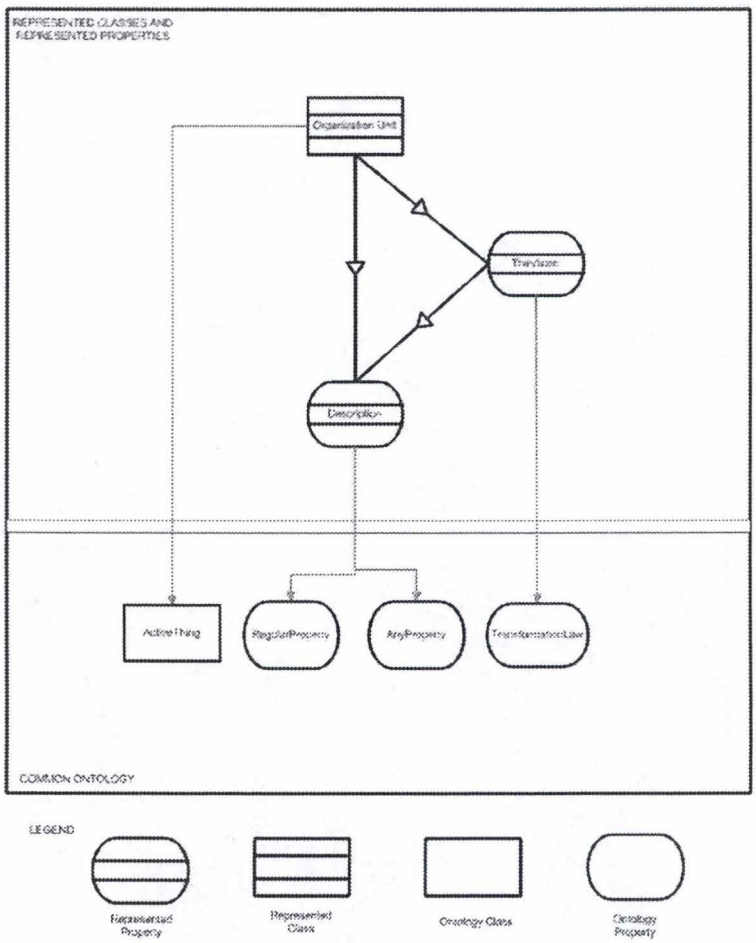

Figure A.40: Ontological mapping of BMM Vision

A.41 BMM Weakness

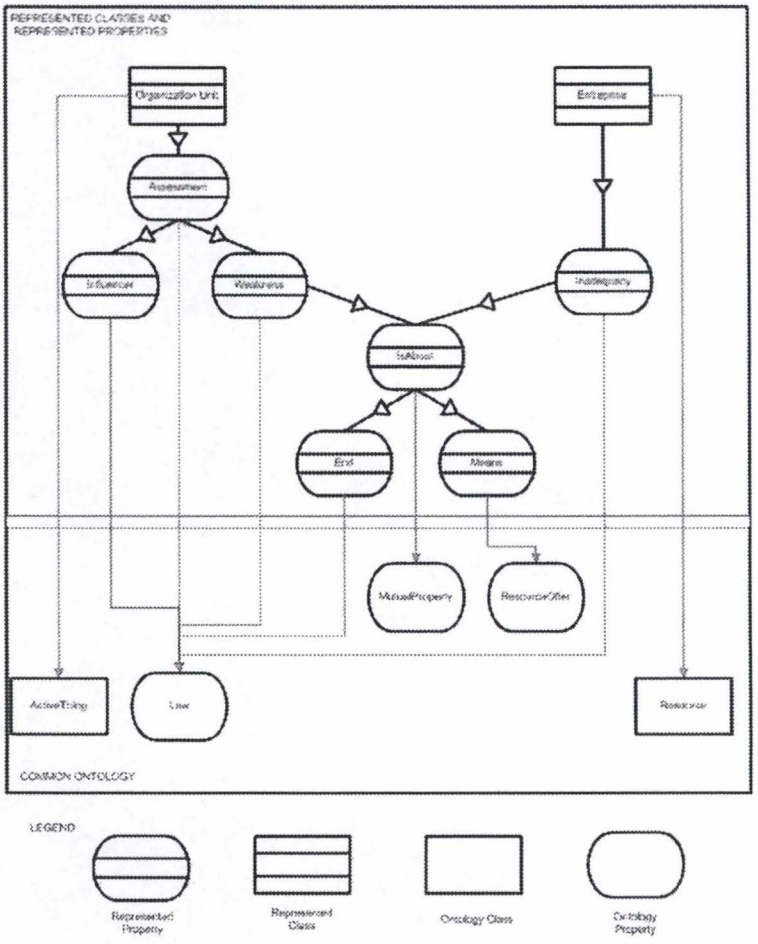

Figure A.41: Ontological mapping of BMM Weakness

Appendix B

*i** Mappings

B.1 *i** Actor Association Link

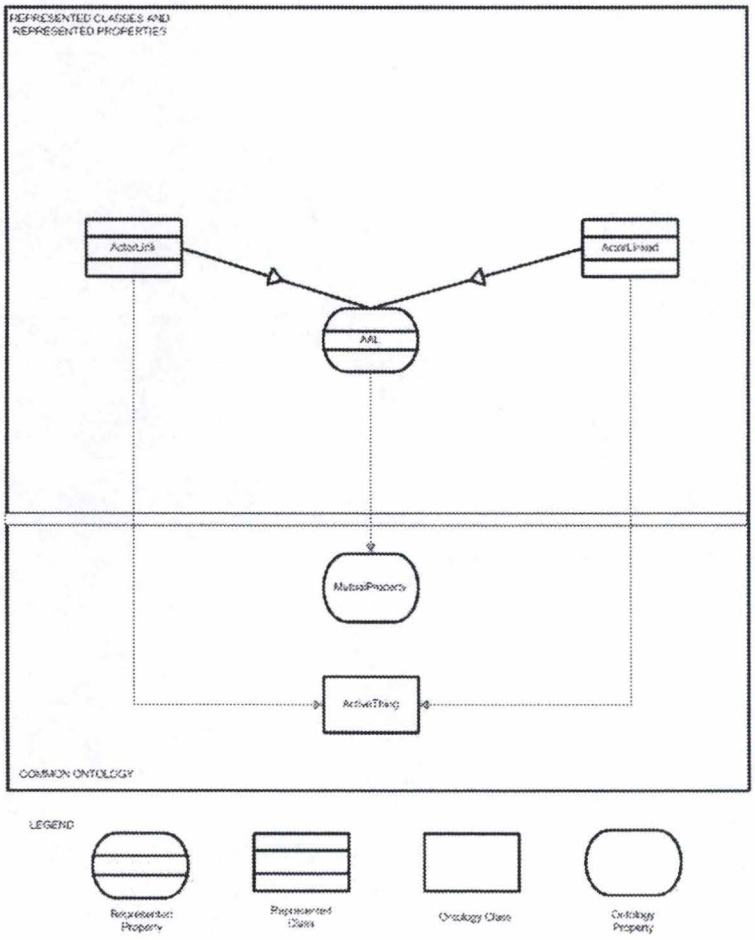

Figure B.1: Ontological mapping of *i** Actor Association Link

B.2 i^* Agent

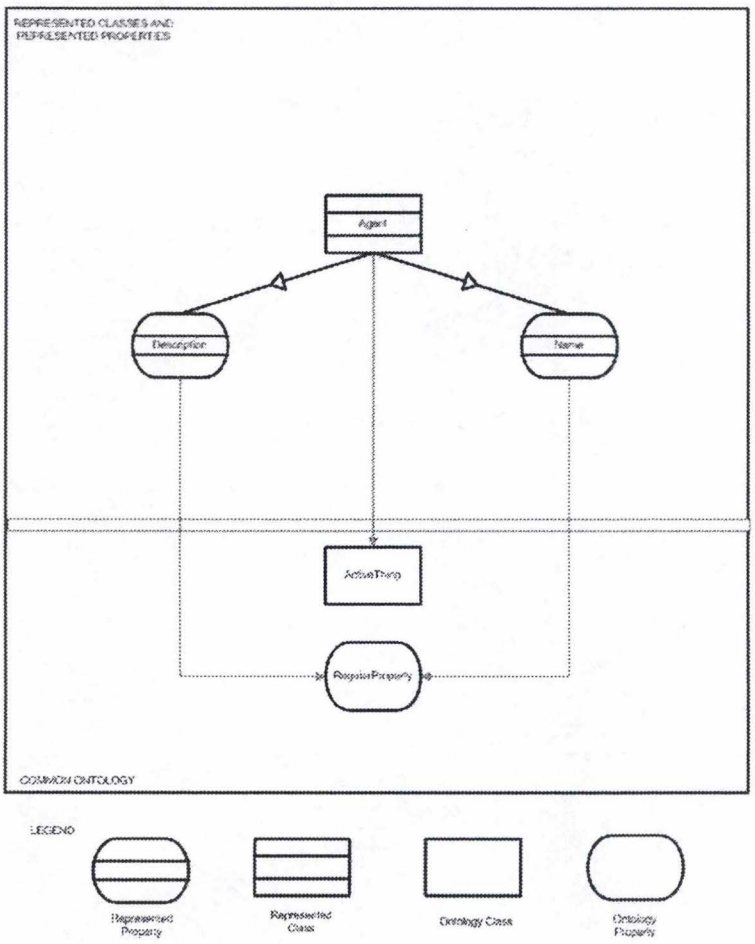

Figure B.2: Ontological mapping of i^* Agent

B.3 i^* Belief

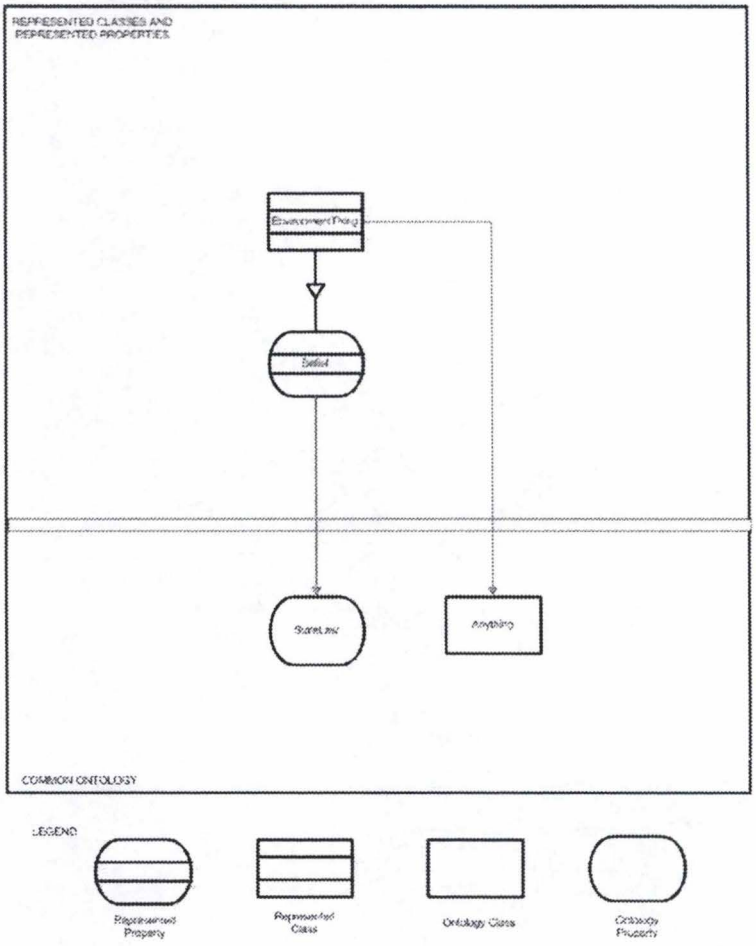

Figure B.3: Ontological mapping of i^* Belief

B.4 i^* Contribution Link

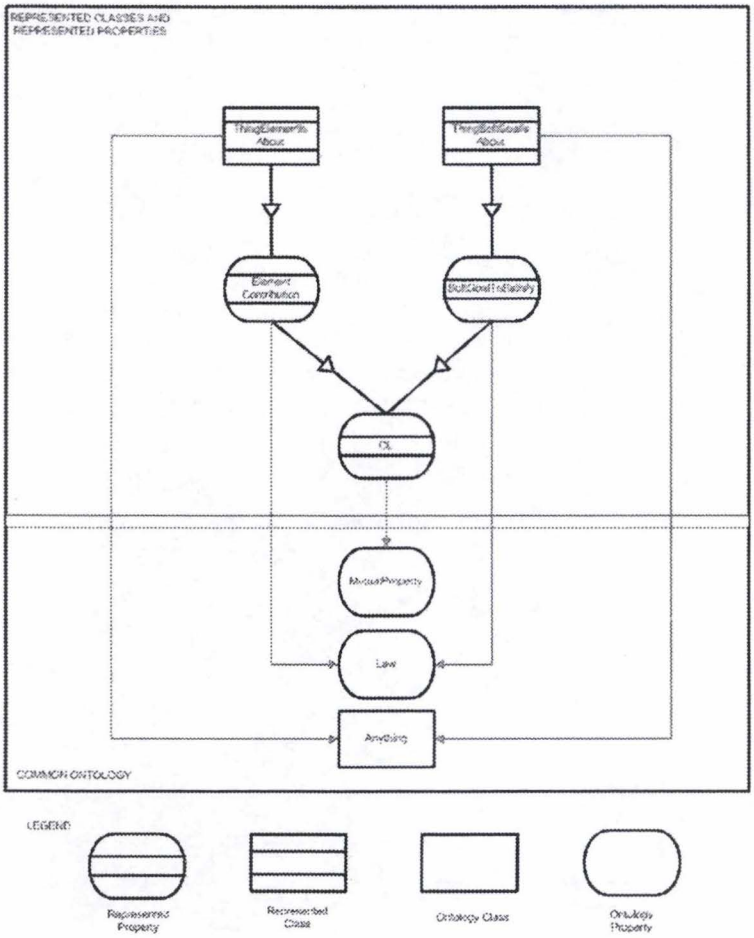

Figure B.4: Ontological mapping of i^* Contribution Link

B.5 i^* Decomposition Link

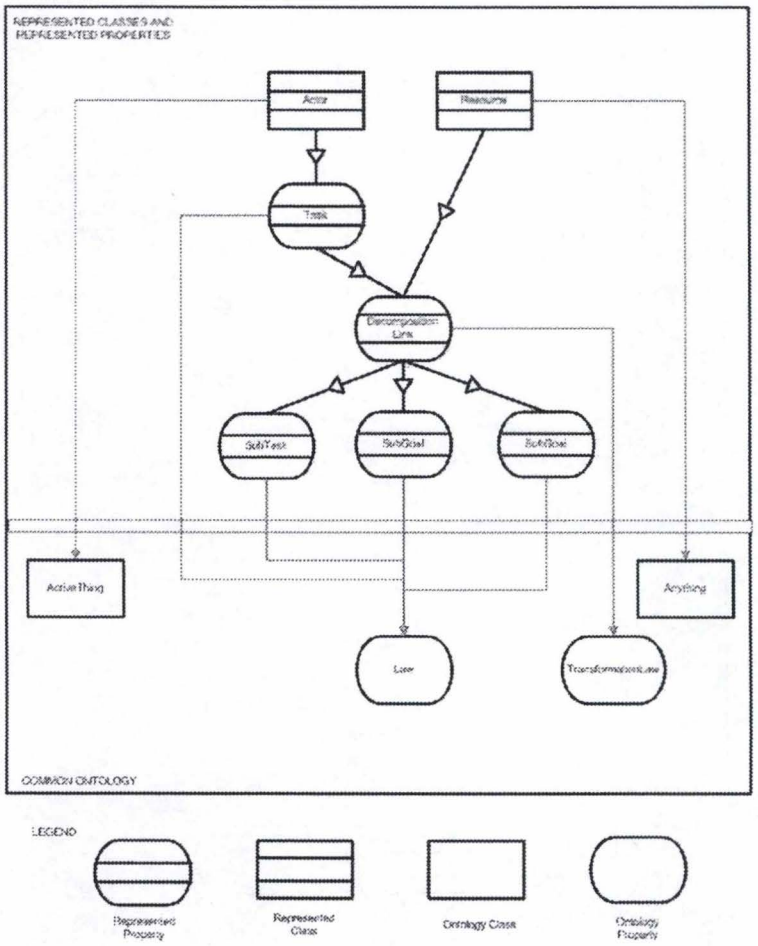

Figure B.5: Ontological mapping of i^* Decomposition Link

B.6 i^* Dependency Link

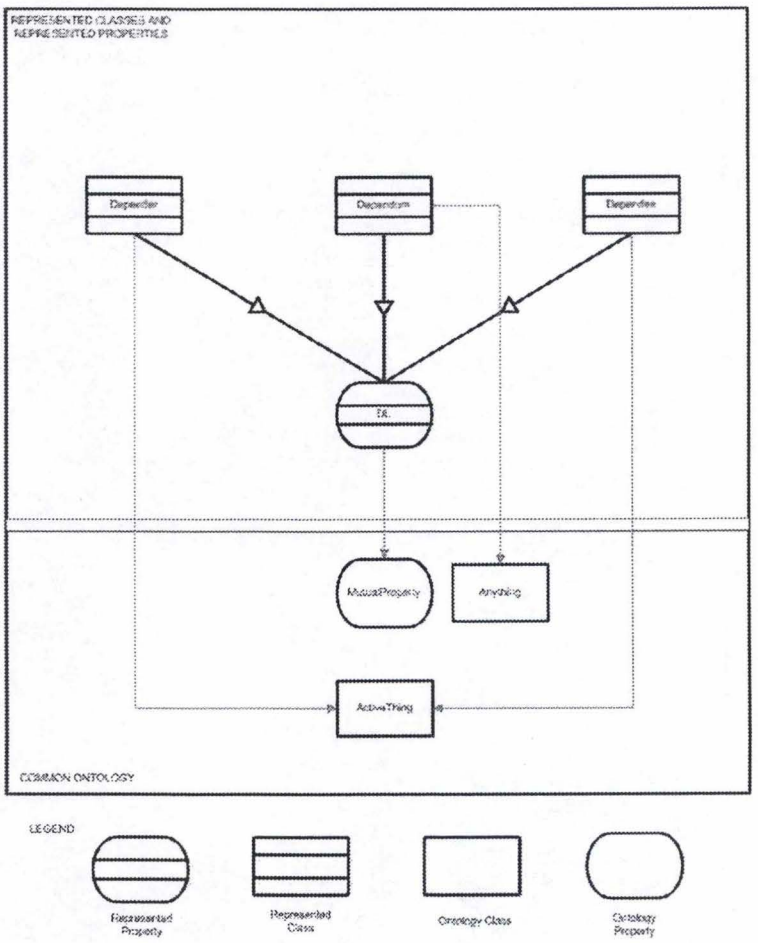

Figure B.6: Ontological mapping of i^* Dependency Link

B.7 i^* Goal

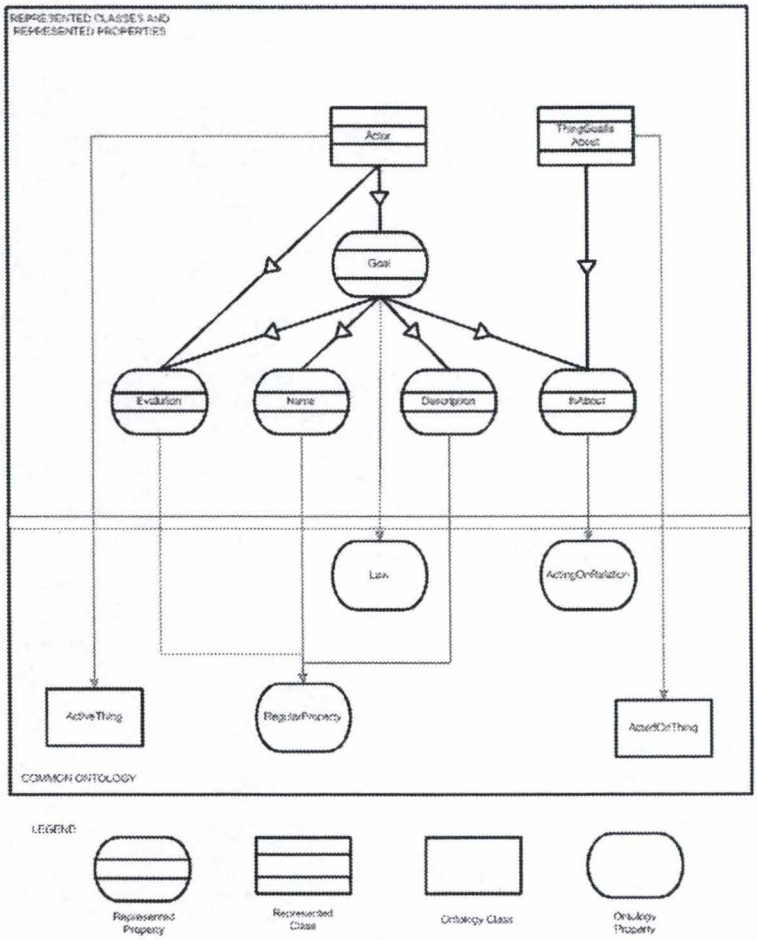

Figure B.7: Ontological mapping of i^* Goal

B.8 i^* Means Ends Link

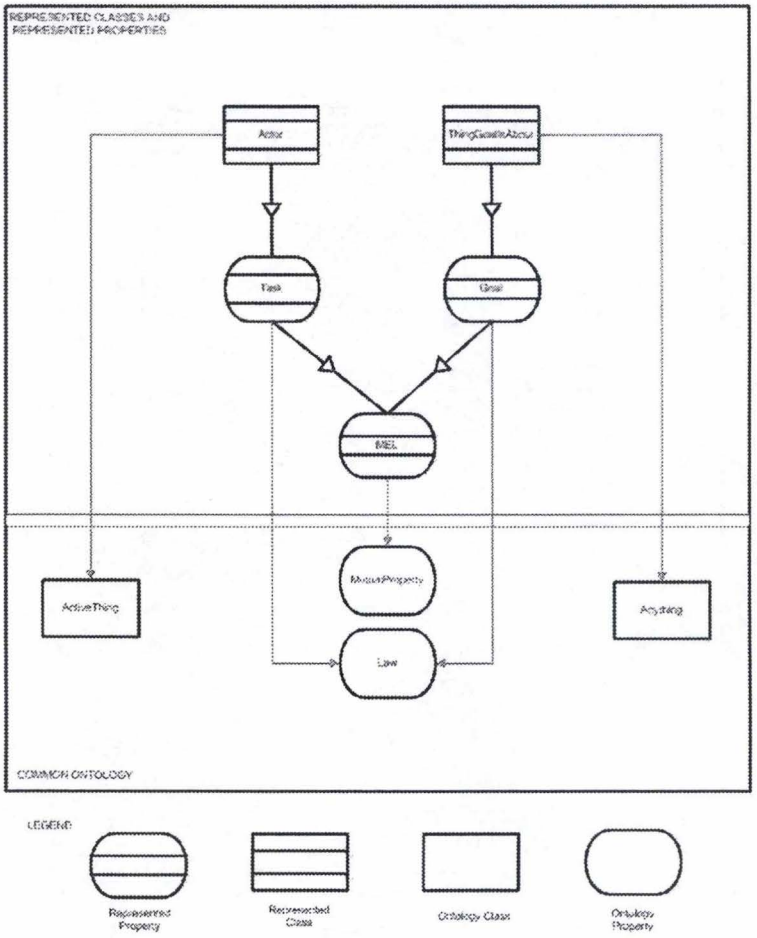

Figure B.8: Ontological mapping of i^* Means Ends Link

B.9 i^* Position

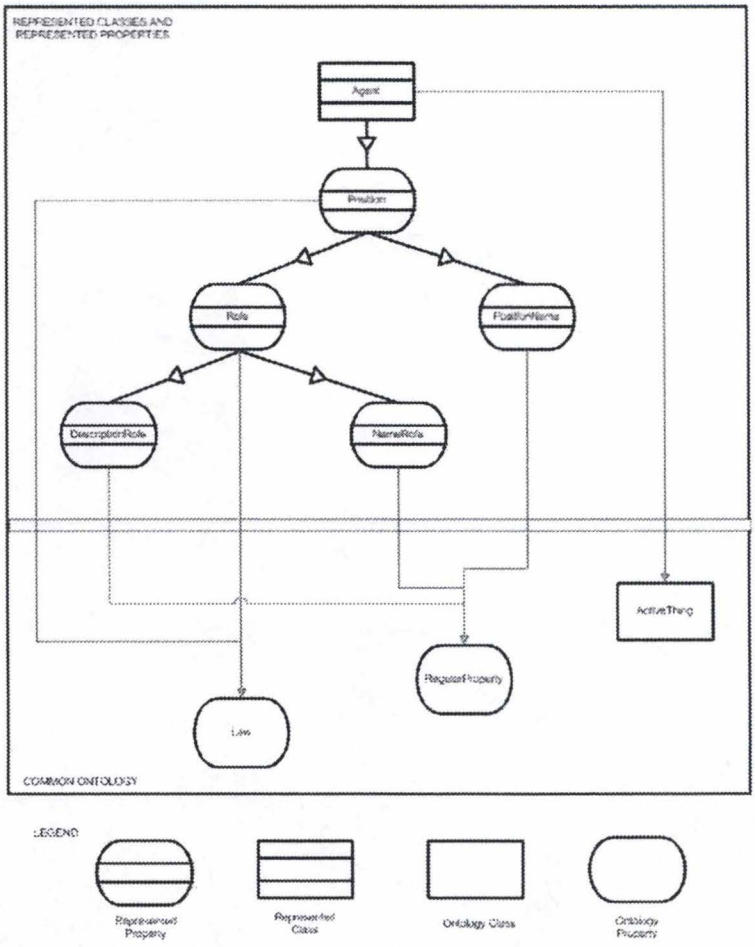

Figure B.9: Ontological mapping of i^* Position

B.10 i^* Resource

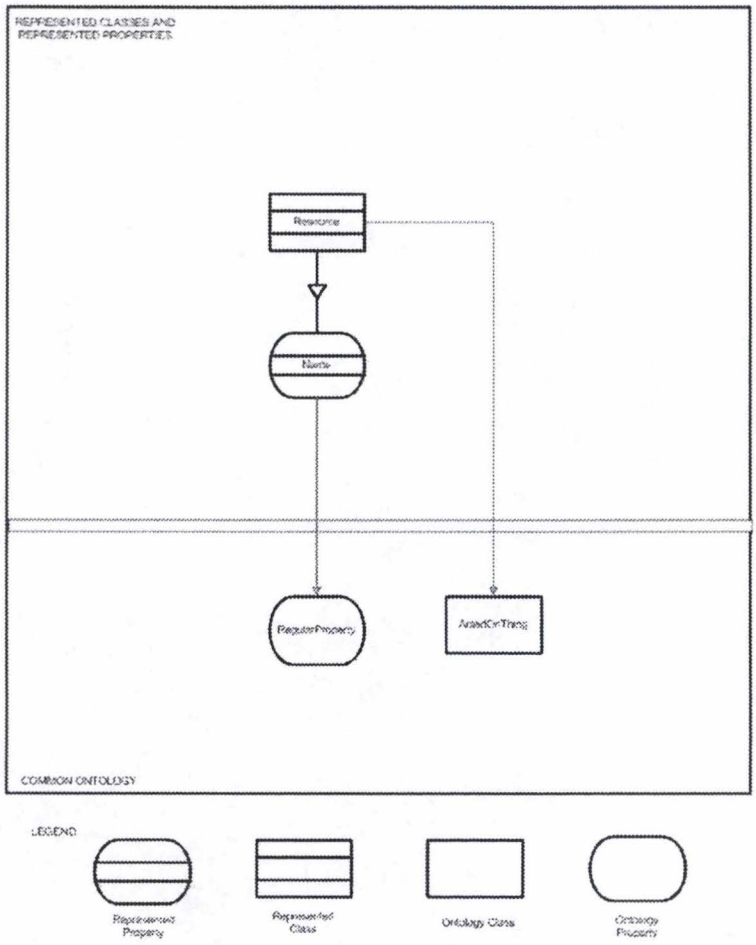

Figure B.10: Ontological mapping of i^* Resource

B.11 *i** Role

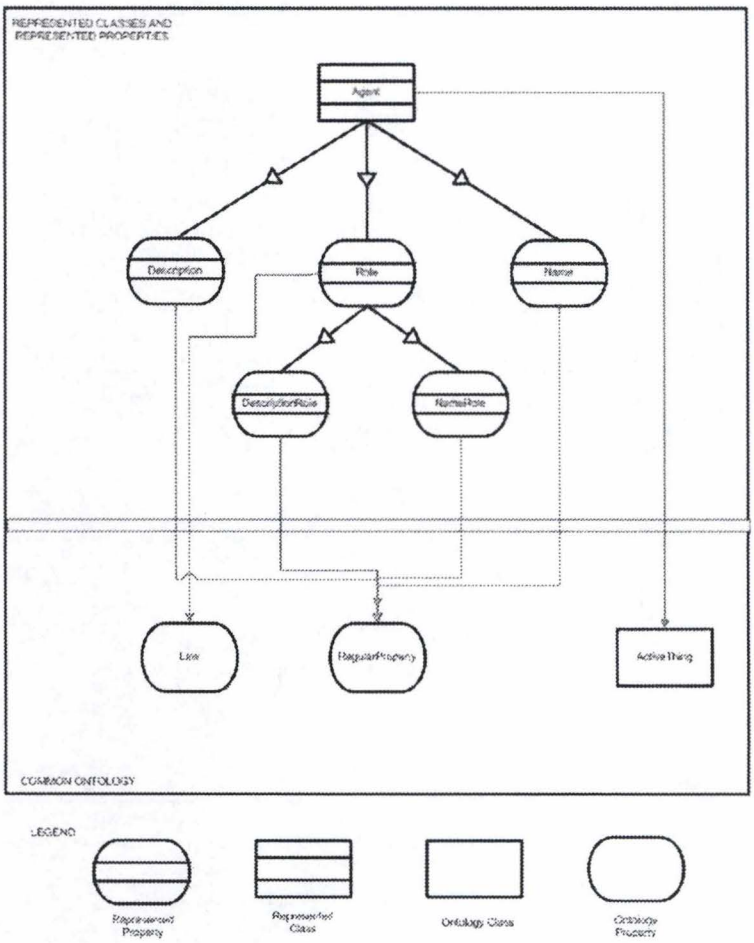

Figure B.11: Ontological mapping of *i** Role

B.12 *i** SoftGoal

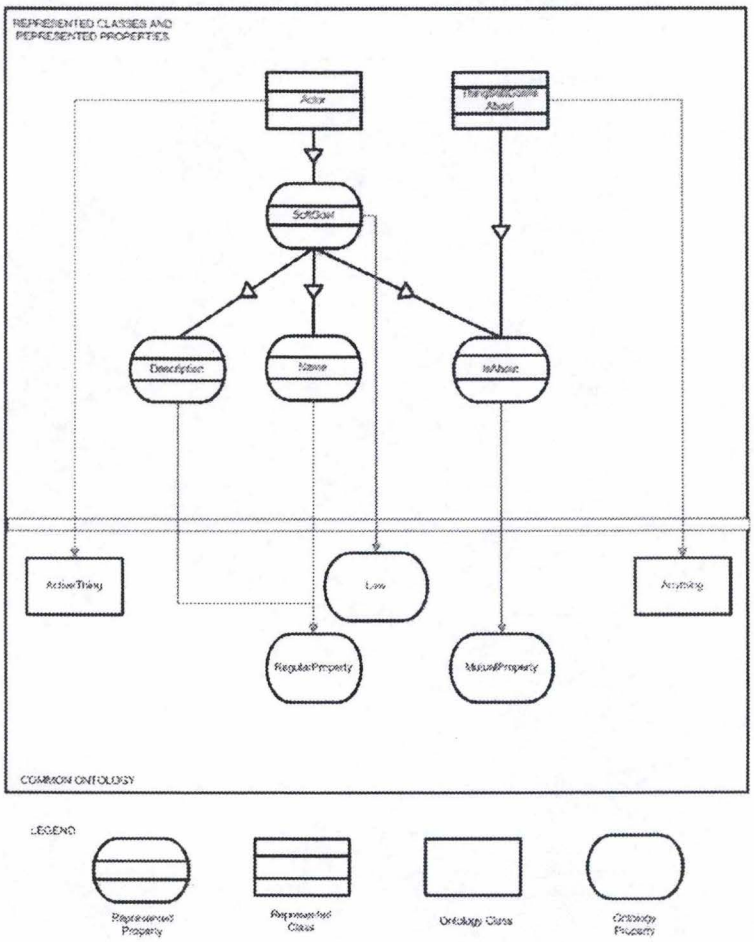

Figure B.12: Ontological mapping of *i** SoftGoal

B.13 *i** Task

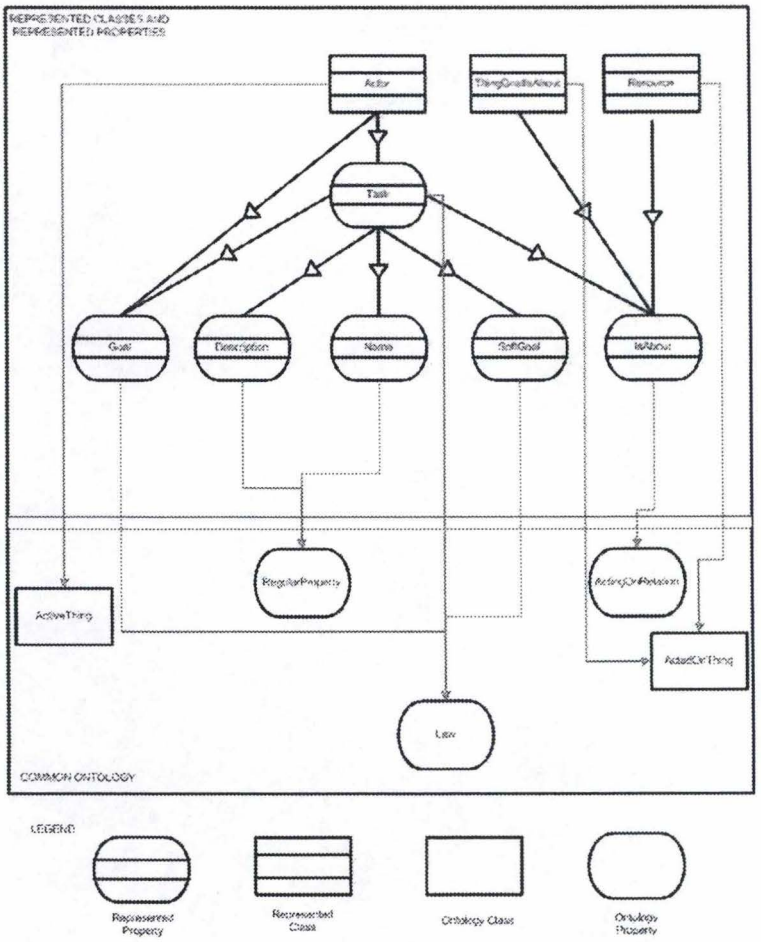

Figure B.13: Ontological mapping of *i** Task